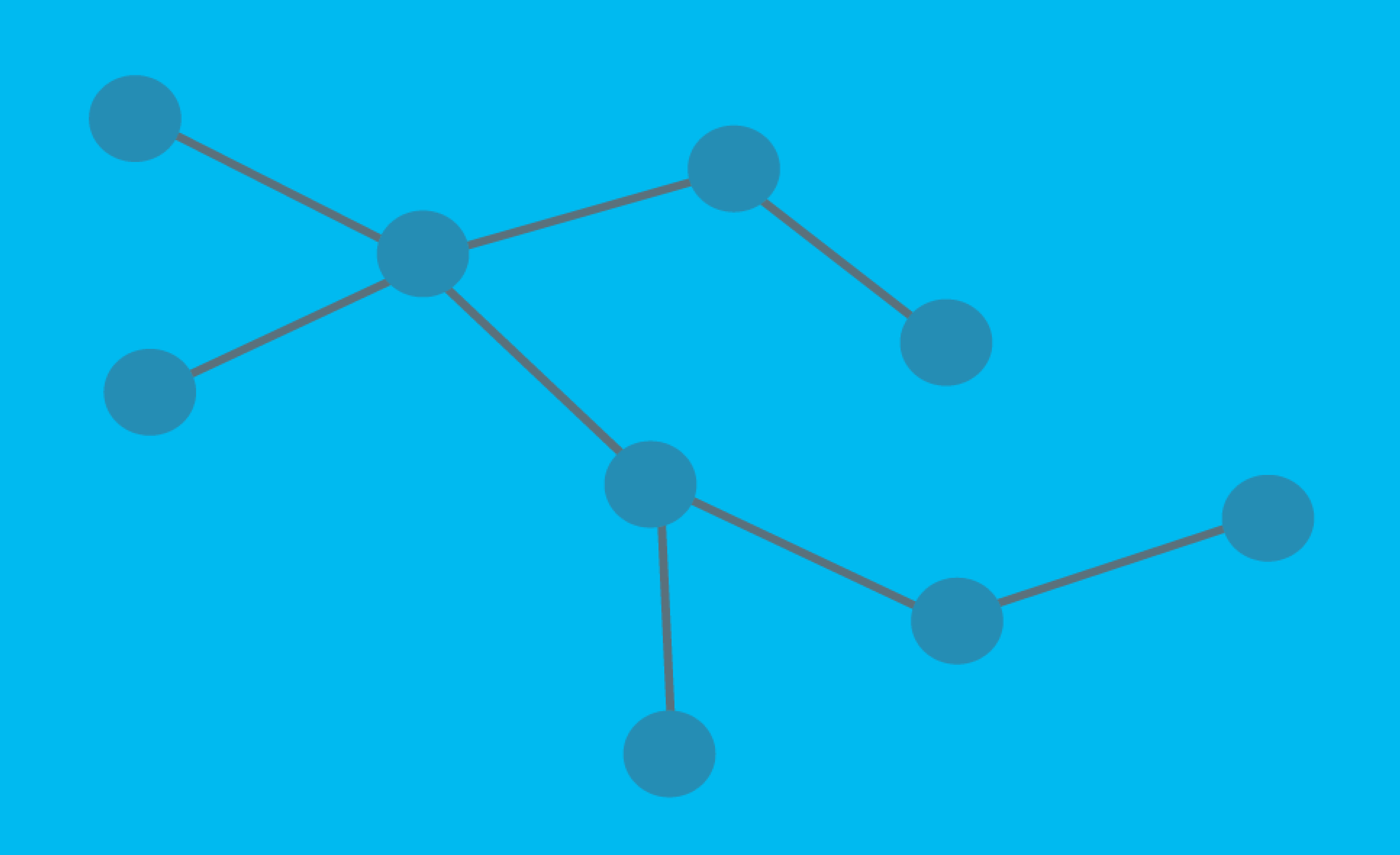
交互式Web前端开发实践

冷亚洪　黄　炜　宋　宇　阚　洪　李发陵　编　著



清华大学出版社

交互式 Web 前端开发实践

冷亚洪　黄　炜　宋　宇　阚　洪　李发陵　编著

清华大学出版社
北　京

内 容 简 介

本书以 Web 前端开发过程中的实际需要和应该掌握的技术为基础，全面、系统地介绍了 Web 前端开发所涉及的相关知识点和开发技巧，涵盖 HTML（含 HTML 5）、CSS（含 CSS 3）、JavaScript 基础及 jQuery 框架等网页设计基础知识、高级编程知识。每章都配置了大量的实用案例，图文并茂，效果直观。

本书分 3 部分，共 8 章。第一部分为基础篇，主要介绍 Web 前端开发基础知识、HTML 标记语言、HTML 5 新特性、CSS 基础知识、盒子模型、CSS+DIV 布局、JavaScript 语言及网页设计方法等内容；第二部分为进阶篇，详细介绍了 JavaScript 的面向对象编程思想及常用的 JavaScript 框架，重点介绍了 jQuery 框架的使用，并辅以大量的案例和综合实例进行讲解，让读者能通过本阶段的学习提高前端设计和编程的能力；第三部分为实战篇，综合运用前两部分的理论知识，结合软件开发流程，详细讲解了"点餐系统"前端的功能设计、编程实现及各方面的内容和技巧。

本书可作为从事 Web 前端开发、网页设计与制作、网站开发及网页编程等行业人员的参考书，也可作为应用型本科院校及培训学校计算机及相关专业的教材。

图书在版编目（CIP）数据

交互式 Web 前端开发实践/冷亚洪等编著. —北京：清华大学出版社，2017
ISBN 978-7-302-47171-4

Ⅰ. ①交…　Ⅱ. ①冷…　Ⅲ. ①网页制作工具—程序设计　Ⅳ. ①TP393.092.2

中国版本图书馆 CIP 数据核字（2017）第 116561 号

责任编辑：张龙卿
封面设计：徐日强
责任校对：袁　芳
责任印制：李红英

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn，010-62770175-4278
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：20.25　　**字　　数**：486 千字
版　　次：2017 年 6 月第 1 版　　**印　　次**：2017 年 6 月第 1 次印刷
印　　数：1～2000
定　　价：39.80 元

产品编号：073397-01

前　言

近十年来，信息技术产业发展态势良好，在我国经济发展中起着非常重要的推动作用。信息技术产业也是“十三五”期间被列为重点发展的产业，市场前景广阔，Web 前端开发工程师的需求量大幅增加。一名合格的 Web 前端开发工程师必须掌握基本的 Web 前端开发技术，包括 CSS、HTML、DOM、JavaScript、ajax 等，在掌握这些技术的同时，更要清楚地了解它们在不同浏览器上的兼容情况、渲染原理和潜在的 Bug。一名合格的前端工程师除了掌握网站性能优化、SEO 和服务器端的基础知识等知识结构之外，还必须学会运用各种工具进行辅助开发，比如处理 IE 系列浏览器兼容性问题的 IEtester，Firefox 排错用的 Firebug、FlashFirebug、Debugger 等调试工具。总而言之，一名合格的 Web 前端开发工程师不但要掌握技术层面的知识，还要掌握理论层面的知识，包括代码的可维护性、组件的易用性、分层语义模板和浏览器分级支持等内容。

本书基于 HTML、CSS、jQuery、ajax 等技术编写，重点突出交互式的 Web 前端技术实现。本书囊括了编者及其团队成员多年 Web 前端开发与设计的经验，是一本可以使读者快速建立规范的 Web 前端开发意识和工程化软件开发思想的书籍，是一本可以使读者快速提高 Web 前端开发技能并快速达到 Web 前端开发工程师岗位任职能力要求的书籍。本书内容编排结构合理，知识由浅入深，以较全面的知识点、丰富的案例、完整的综合项目实践为主要内容，结合分层开发思想，循序渐进地引导读者在基础篇学习基础理论，在进阶篇学习编程技巧，在高级篇通过综合项目实战提高 Web 前端开发技能。

本书由冷亚洪、黄炜负责全书的编写、统稿、知识点及案例设计。具体编写分工为：第 1 章由李发陵编写，第 2 章由冷亚洪编写，第 3、4 章由阚洪编写，第 5、6 章由宋宇编写，第 7、8 章由黄炜编写。

本书的特色如下：

（1）本书内容编排结构合理，知识点由浅入深，循序渐进地引导读者快速入门，并能提高初级及以上读者的实际应用水平，让读者能够快速适应对 Web 前端开发工程师岗位的新要求。

（2）本书采用“案例制”和“项目制”相结合的思想，通过大量的案例帮助读者对知识点的理解及掌握，使用综合项目案例（在线订餐系统）提升读者的综合应用能力。

(3) 本书重点突出 Web 的交互式开发,结合软件的面向对象和分层开发思想进行编程,让读者掌握的不仅是传统的 HTML+DIV+CSS+jQuery 编程,而且会掌握软件开发思想,掌握前端与后台之间的交互式设计与实现。

(4) 本书在综合项目案例部分,提供了 Java 和 C# 两种语言实现后台的数据处理,方便 Java 和 C# 方向的读者学习。

(5) 本书使用市场上最流行的软件开发技术,使读者在完成本书的学习后,可以无缝地过渡到对应的工作岗位。

我们期望本书能为阅读者们提供以下帮助。

(1) 软件类应用型本科或高职高专人才培养的实训教材。

(2) Web 前端开发工程师的岗前学习教材。

(3) 培训机构的培训教材。

(4) Web 前端开发工程师的能力提升学习书籍。

在本书的编写过程中,参阅了大量的资料,尤其是参考文献中列出的资料。在此对所有资料的编写者表示衷心的感谢!由于本书内容涉及面广,加之编者的水平有限,不当之处在所难免,恳请广大读者朋友批评、指正,我们将不胜感激,编者的邮箱是 7357220@qq.com。

编　者

2017 年 1 月

目　录

第一部分　基　础　篇

第二部分　进　阶　篇

第三部分　实　战　篇



第一部分

基　础　篇



第 1 章　Web 前端开发概述

1.1　Web 概述

Web 是 Internet 中最受欢迎的一种多媒体信息服务系统。整个系统由 Web 服务器、浏览器和通信协议组成。通信协议 HTTP 能够传输任意类型的数据对象来满足 Web 服务器与客户之间的多媒体通信的需要。Web 带来的是世界范围的超级文本服务。用户可通过 Internet 从全世界任何地方调来所希望得到的文本、图像(包括活动影像)和声音等信息。另外,Web 还可提供其他的 Internet 服务如 Telnet、FTP、Gopher 和 Net User 等。

在 Web 网站上,不仅可以传递文字信息,还可以传递图形、声音、影像、动画等多媒体信息。Web 的成功在于使用了 HTTP 超文本传输协议,制定了一套标准的、易为人们掌握的超文本标记语言 HTML,使用了信息资源的统一定位格式 URL。我们可以把 Web 看作一个图书馆,而每一个网站就是这个图书馆中的一本书。每个网站都包括许多画面,进入该网站时显示的第一个画面就是"主页"或"首页"(相当于书的目录),而同一个网站的其他画面都是"网页"(相当于书页)。

1. HTTP 协议

从网络协议的角度看,HTTP 是对 TCP/IP 协议集的扩展,作为浏览器与服务器间的通信协议,处于 TCP/IP 层次中的应用层。

HTTP 是一种无状态协议,即服务器不保留与客户交易时的任何状态,这可以大大减轻服务器的存储负担,从而保持较快的响应速度。HTTP 又是一种面向对象的协议,允许传输任意类型的数据对象。它通过数据类型和长度来标识所传送的数据内容和大小,并允许对数据进行压缩传送。浏览器软件配置于用户端计算机上,用户发出的请求通过浏览器分析后,按 HTTP 规范送给服务器,服务器按用户需求,将 HTML(超文本标记语言)文档送回给用户。

2. Web 服务的基本过程

Web 最吸引人的地方是它的"简单性",其工作过程也是客户机/服务器模式(C/S)。Web 的工作可分为 4 个基本阶段:连接、请求、响应和关闭(见图 1-1)。它们都属于 HTTP 的下层基础。信息资源以网页(HTML 文件)形式存储在 Web 服务器中,当用户希望得到某种信息时,要先与 Internet 沟通连接(上网);然后用户通过 Web 客户端程序(浏览器)向 Web 服务器发出请求;Web 服务器根据客户的请求给予响应,将在 Web 服务器中存放的、符合用户要求的某个网页发送给客户端,浏览器在收到该页面后对其进行解释,最终将图文等信息呈现给客户;一次 Web 服务操作结束后,关闭此次连接,或用户根据需要进行下一次

请求。这样，用户可以通过网页中的链接，方便地访问位于其他 Web 服务器中的页面或其他类型的网络信息资源。

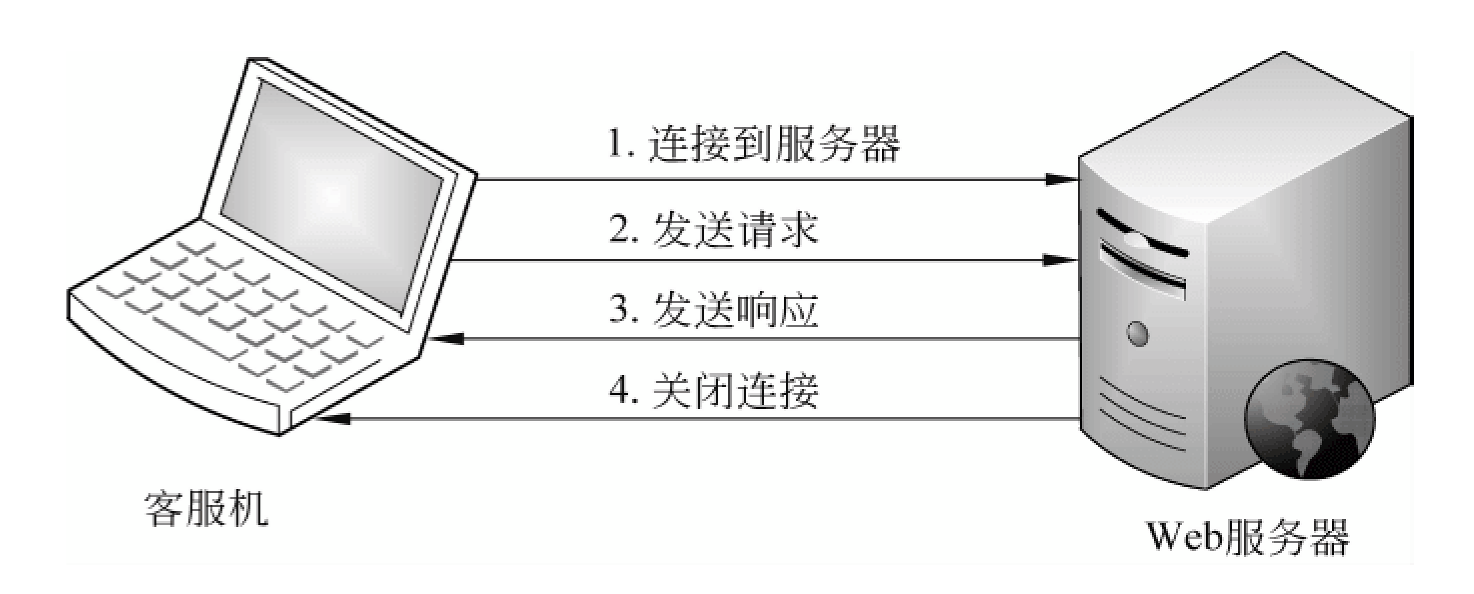


图 1-1　HTTP 的请求响应模型

Web 服务器集成了所有视觉辅助效果来表示信息，这些信息可以按多种格式存在，易于浏览和理解。例如，在讨论复杂问题时，可以使用图表、影像剪辑甚至交互式应用程序，而不仅仅是字符文本，这样便于解释论题，使人一目了然。与其他信息发布工具相比，Web 服务由于所需的费用很低并且覆盖面广，因而具有很大的吸引力。另外，使用各种搜索机制和 Web 站点分类目录数据库注册一个 Web 站点，可以使客户在需要时得到所需的信息。

1.1.1　Web 的发展

最早的网络构想可以追溯到 1980 年蒂姆·伯纳斯·李构建的 ENQUIRE 项目。这是一个类似维基百科的超文本在线编辑数据库。尽管这与万维网大不相同，但是它们有许多相同的核心思想，甚至还包括一些伯纳斯·李的万维网（World Wide Web，WWW，也作 Web、W3）之后的下一个项目语义网中的构想。

1989 年 3 月，伯纳斯·李撰写了《关于信息化管理的建议》一文，文中提及 ENQUIRE 并且描述了一个更加精巧的管理模型。1990 年 11 月 12 日他和罗伯特·卡里奥（Robert Cailliau）合作提出了一个更加正式的关于万维网的建议。在 1990 年 11 月 13 日他在一台工作站上设计了第一个网页以实现他文中的想法。

在那年的圣诞假期，伯纳斯·李制作了让一个网络工作所必需的所有工具：第一个万维网浏览器（同时也是编辑器）和第一个网页服务器。

1991 年 8 月 6 日，他在 alt.hypertext 新闻组上发表了万维网项目简介的文章，这一天也标志着因特网上万维网公共服务的首次亮相。

万维网中至关重要的超文本概念起源于 20 世纪 60 年代的几个项目。譬如泰德·尼尔森（Ted Nelson）的仙那都项目（project Xanadu）和道格拉斯·英格巴特（Douglas Engelbart）的 NLS。而这两个项目的灵感都是来源于万尼瓦尔·布什在其 1945 年发表的《和我们想的一样》论文中为微缩胶片设计的“记忆延伸”（memex）系统。

蒂姆·伯纳斯·李的另一个才华横溢的突破是将超文本嫁接到因特网上。在他的《编织网络》一书中，他解释说自己曾一再向使用者们建议这两种技术的结合是可行的，但是没有任何人响应他的建议，最后他只好自己解决了这个问题。他发明了一个全球网络资源唯一认证的系统：统一资源标识符。

万维网和其他超文本系统有很多不同之处。

(1) 万维网上需要单项链接而不是双向链接，这使得任何人可以在资源拥有者不做任何改变时链接该资源。与早期的网络系统相比，这对于网络服务器和网络浏览器来说已经是很大的进步，但它的副作用是产生了坏链的问题。

(2) 万维网不像某些应用软件如 HyperCard 那样是私有的，这使得服务器和客户端能够独立地发展和扩展，而不受许可权限制。

1993 年 4 月 30 日，欧洲核子研究组织宣布万维网对任何人免费开放，并且不收取任何费用。两个月之后 Gopher 宣布不再免费，从而造成大量用户从 Gopher 转向万维网联盟。万维网联盟(World Wide Web Consortium，W3C)又称 W3C 理事会。1994 年 10 月在麻省理工学院计算机科学实验室成立，建立者是万维网的发明者蒂姆·伯纳斯·李。

1.1.2　Web 特点及架构

1. Web 的形式及特点

(1) 易于导航的图形化 Web

Web 非常流行的一个很重要的原因就在于它可以在一页上同时显示色彩丰富的图形和文本的性能。在 Web 之前 Internet 上的信息只有文本形式。Web 可以提供将图形、音频、视频信息集合于一体的特性。同时，Web 是非常易于导航的，只需要从一个链接跳到另一个链接，就可以在各页各站点之间进行浏览。

(2) 与平台无关的 Web

无论用户的系统平台是什么，都可以通过 Internet 访问万维网。浏览万维网对用户的系统平台没有什么限制。对万维网的访问是通过一种叫作浏览器(browser)的软件实现的，如 Netscape 的 Navigator、NCSA 的 Mosaic、Microsoft 的 Internet Explorer 等。

(3) 分布式的 Web

大量的图形、音频和视频信息会占用相当大的磁盘空间，用户甚至无法预知信息的多少。对于 Web 没有必要把所有信息都放在一起，信息可以放在不同的站点上。只需要在浏览器中指明这个站点就可以了。这使得在物理上并不一定在一个站点的信息在逻辑上实现了一体化，至少从用户的角度来看这些信息是一体的。

图 1-2 展示了 Web 之间的典型链接方式，Web 站点都存在于不同的物理位置，站点上存放着各种文档，这些文档中有一些文字与其他文字的显示方式有所区别，用于链接到其他站点，称为“超链接”，用户只要在上面单击，浏览器就可以跳转到对应的站点并显示相应的内容。

(4) 交互式的动态 Web

由于各 Web 站点的信息包含站点本身的信息，信息的提供者可以经常对站点中的信息进行更新，如某个协议的发展状况、公司的广告等，Web 站点上的信息是需要经常更新的。

Web 的交互性首先表现在它的超链接上，用户的浏览顺序和所访问的站点完全由他自己决定。系统另外通过 HTTP 的 Get 请求从服务器中获得动态的信息，用户通过填写 form 表单向服务器提交(HttpPost)请求，服务器根据用户的请求返回相应信息，实现良好的人机交互，如图 1-3 所示。

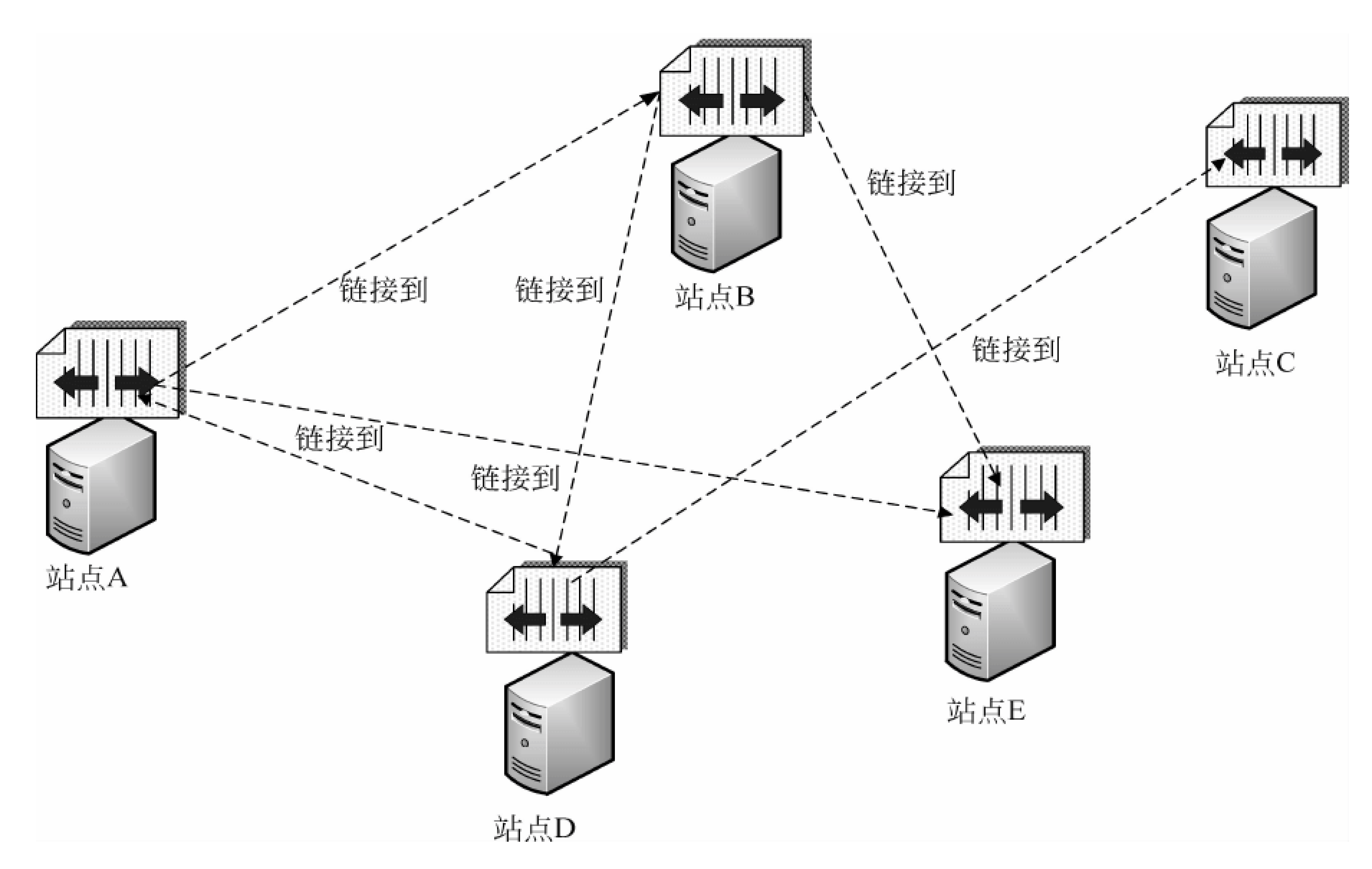


图 1-2　Web 站点之间的链接

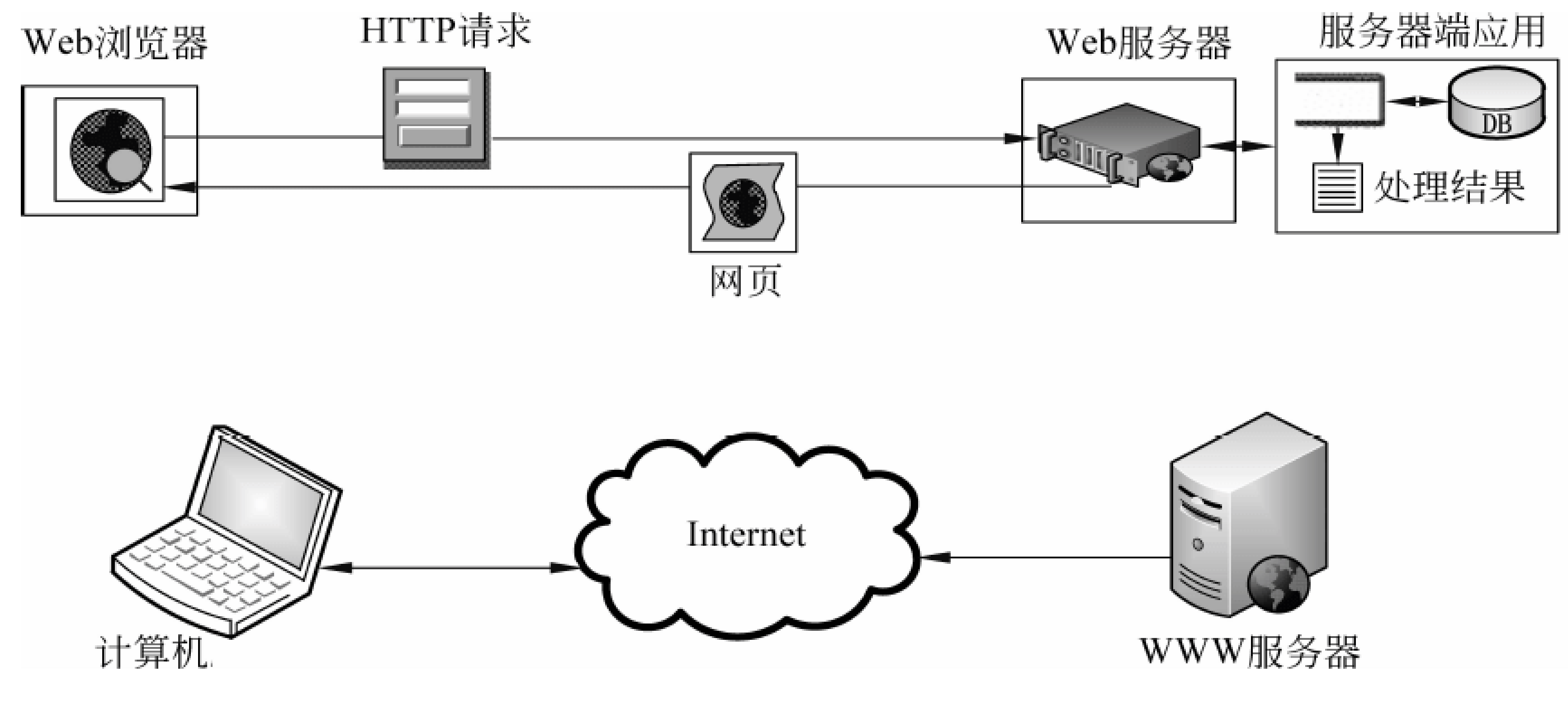


图 1-3　动态网页示意图

2. C/S 与 B/S 架构

(1) C/S 架构

C/S(Client/Server)又称客户/服务器模式，是大家熟知的软件系统体系结构，这种模式通过将任务合理分配到客户端和服务器端，降低了系统的通信开销，但需要安装客户端才可进行管理操作。

客户端和服务器端的程序不同，用户的程序主要在客户端，客户端程序主要完成用户具体的业务。客户端需要安装专用的客户端软件。服务器端主要提供数据管理、数据共享、数据及系统维护和并发控制等功能。服务器通常采用高性能的 PC、工作站或小型机，并采用大型数据库系统，如 Oracle、Sybase、Informix 或 SQL Server。

C/S 模式的优点是能充分发挥客户端 PC 的处理能力，很多工作可以在客户端处理后再提交给服务器，其优点就是客户端响应速度快，存在的缺点如下：

随着互联网的飞速发展，移动办公和分布式办公越来越普及，这需要使我们设计的系统具有扩展性。这种方式的远程访问需要专门的技术，同时要对系统进行专门的设计来处理分布式的数据。

客户端需要安装专用的客户端软件。首先涉及安装的工作量；其次任何一台计算机出问题，如病毒、硬件损坏，都需要进行安装或维护，特别是一个单位有很多分部或专卖店的情况下，不是工作量的问题，而是路程的问题。还有，系统软件升级时，每一台客户机都需要重新安装，其维护和升级成本非常高。

客户端的操作系统一般有一定的限制。比如使用 Windows 98 的客户端不能用 Windows 2000 或 Windows XP，或者不适用于微软新的操作系统等，更不用说 Linux、UNIX 等。

（2）B/S 架构

B/S（Browser/Server）架构是浏览器和服务器结构（见图 1-4）。它是随着 Internet 技术的兴起而对 C/S 架构进行改进的结构。

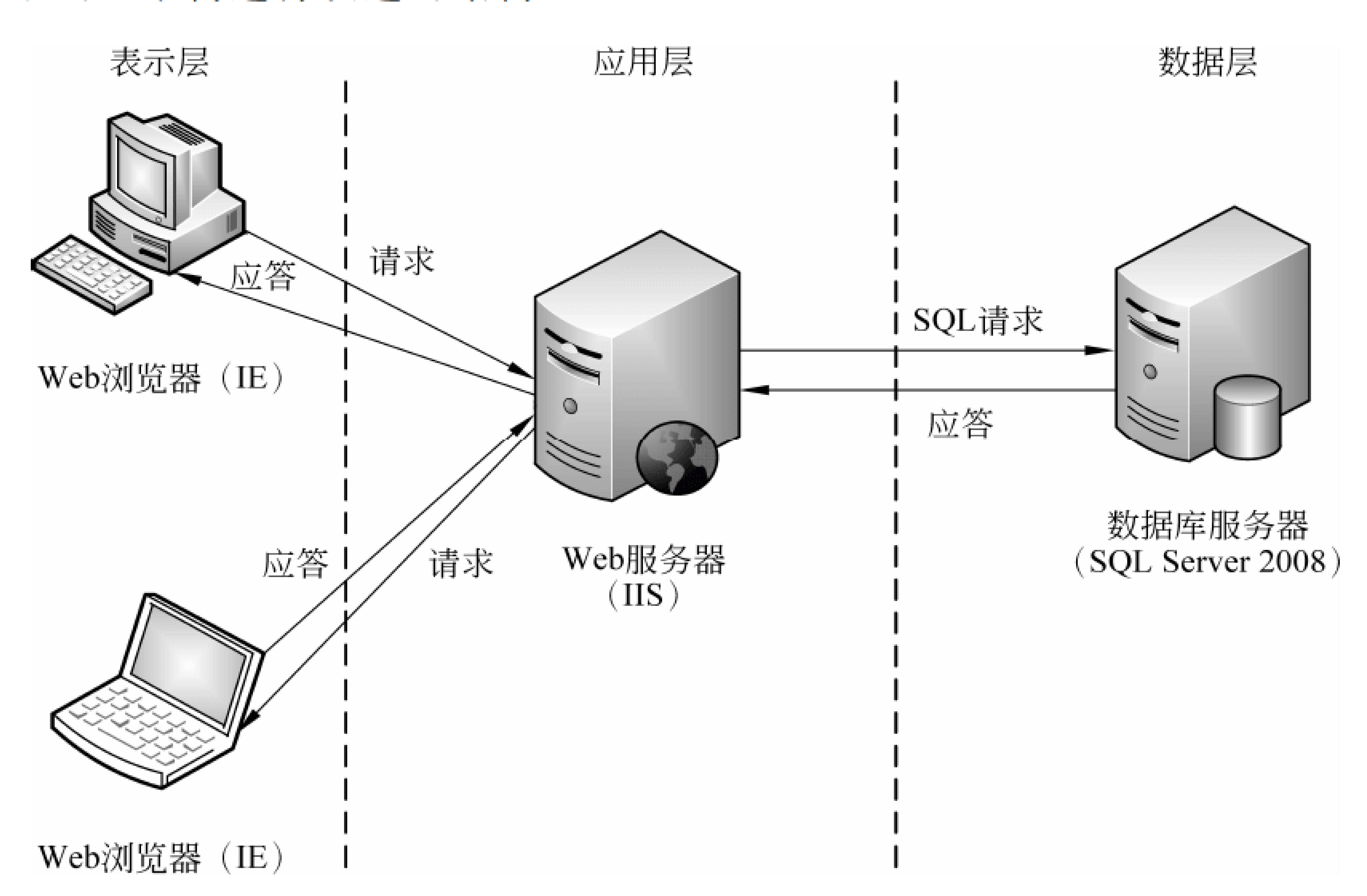


图 1-4　B/S 系统架构

这种架构中客户端（Browser）几乎没有专门的应用程序，应用程序基本上都在服务器端（Server）。由于客户端没有程序，应用程序的升级和维护都可以在服务器端完成，升级维护十分方便。由于客户端使用浏览器，使得用户界面“丰富多彩”，但数据的打印输出等功能受到了限制。为了克服这个缺点，一般把利用浏览器方式实现困难的功能单独开发成可以发布的控件，再在客户端利用程序进行调用。

B/S 结构是 Web 兴起后的一种网络结构模式，Web 浏览器是客户端最主要的应用软件。这种模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用。客户机上只要安装一个浏览器（Browser），如 Netscape Navigator 或 Internet Explorer，服务器上安装 Oracle、Sybase、Informix 或 SQL Server 等数据库。浏览器通过 Web Server 同数据库进行数据交互，这样就大大简化了客户端计算机的载荷，减轻

了系统维护与升级的成本和工作量，降低了用户的总体成本(TCO)。

(3) C/S 与 B/S 的区别

C/S 建立在局域网的基础上，B/S 建立在广域网的基础上。其区别主要有如下几点。

① 硬件环境不同：C/S 一般建立在专用的网络上，形成小范围的局域网环境，局域网之间通过专门的服务器提供连接和数据交换服务。B/S 建立在广域网之上，不必限定专门的网络硬件环境。例如电话上网、租用设备、信息管理，有比 C/S 更强的适应范围，一般只要有操作系统和浏览器就行。

② 对安全性的要求不同：C/S 一般面向相对固定的用户群，对信息安全的控制能力很强，一般高度机密的信息系统采用 C/S 结构较适宜，可以通过 B/S 发布部分公开信息。B/S 建立在广域网之上，对安全的控制能力相对弱，面向不可知的用户群。

③ 程序的架构不同：C/S 程序可以更加注重流程，可以对权限进行多层次校验，对系统的运行速度可以较少考虑。B/S 对安全性以及访问速度要进行多重考虑，建立在需要更加优化的基础上，比 C/S 有更高的要求。B/S 结构的程序架构是发展的趋势，从微软的 .NET系列到 BizTalk 2000、Exchange 2000 等，全面支持网络构件搭建的系统。SUN 和 IBM 推出的 JavaBean 构件技术等使 B/S 更加成熟。

④ 软件的重用性不同：C/S 程序必须进行整体性的考虑，构件的重用性不如在 B/S 模式下构件的重用性好。B/S 的多重结构，要求构件有相对独立的功能，因此具有相对较好的重用性。

⑤ 系统维护不同：系统维护在软件生存周期中的开销较大。C/S 程序由于整体性强，必须整体考察，处理出现的问题以及系统升级较难，有时可能需要再做一个全新的系统。B/S 程序由构件组成，方便个别构件的更换，可以实现系统的无缝升级，系统维护开销可以减到最小，用户从网上自己下载并安装相关程序就可以实现系统升级。

⑥ 处理的问题不同：C/S 程序可以处理的用户相对固定并且在相同区域，安全性要求与操作系统相关性高。B/S 建立在广域网上，面向不同的用户群且分散在不同地域，这是 C/S 无法做到的，因此与操作系统平台的关系最小。

⑦ 用户接口不同：C/S 大多是建立在 Windows 平台上，表现方法有限。对程序员要求普遍较高。B/S 建立在浏览器上，有更加丰富和生动的方式与用户交流，并且大部分难度较低，可降低开发成本。

⑧ 信息流不同：C/S 程序一般是进行典型的中央集权机械式的处理，交互性相对较低。B/S 信息流向可变化，B2B、B2C、B2G 等信息流向的变化更像一个交易中心。

1.2 Web 新技术的发展及应用

Web 2.0 指的是一个由用户主导来创造、协同合作、分享各种资讯与内容的开放平台。Web 2.0 是网络运用的新阶段，网络成为新的平台，内容因为每位使用者的参与而产生，所产生的个人化的内容通过分享形成了现在 Web 2.0 的世界。Web 2.0 以 Blog、Wiki、SNS 等社交软件的应用为核心。Blog 即网上日记，可以在其中迅速发布想法、与他人交流以及从事其他活动。Wiki(维基百科)是由用户编写内容和共享内容的网站，每个人都可以发表

自己的意见,或者对共同的主题进行扩展或者探讨,由用户共同建设一个大百科全书。SNS(社交网络服务)专指旨在帮助人们建立社会性网络的互联网应用服务。目前,SNS 最有代表性的公司是 Facebook,它也是只提供开放的平台,而不提供具体的内容和服务。

1.2.1　Web 3.0

1. 认识 Web 3.0

Web 3.0 只是由业内人员制造出来的概念词语,最常见的解释是,网站内的信息可以直接和其他网站的相关信息进行交互,能通过第三方信息平台同时对多家网站的信息进行整合使用;用户在互联网上拥有自己的数据,并能在不同网站上使用;完全基于 Web,用浏览器即可实现复杂系统程序才能实现的系统功能。用户数据审计后,同步于网络数据。

Web 3.0 的倡导者和实践者——土豆先生所谈的"什么是 Web 3.0",是目前最好的关于 Web 3.0 的阐述。其实很多人在提到 Web 2.0 的时候就说,还会有 Web 3.0,但他们并不知道 Web 3.0 是什么,会在什么时候实现,以及如何实现,有哪些特点。下面简单阐述一下作者对 Web 3.0 的理解,假如说 Web 1.0 的本质是联合,那么 Web 2.0 的本质就是互动,它让网民更多地参与信息产品的创造、传播和分享,而这个过程是有价值的。Web 2.0 的缺点是没有体现出网民劳动的价值,所以 Web 2.0 很脆弱,缺乏商业价值,需要跟具体的产业结合起来才会获得巨大的商业价值和商业成功。Web 3.0 是在 Web 2.0 的基础上发展起来的,并能够更好地体现网民的劳动价值,且能够实现价值均衡分配的一种互联网方式。

总体而言,Web 3.0 更多的不仅仅是一种技术上的革新,而是以统一的通信协议,通过更加简洁的方式为用户提供更为个性化的互联网信息资讯而定制的一种技术整合,将会是互联网发展中由技术创新走向用户理念创新的关键一步。

互联网的技术日新月异,互联网不断深入人们的生活,Web 3.0 将是彻底改变人们生活的互联网形式。Web 3.0 使所有网上公民不再受到现有资源积累的限制,具有更加平等地获得财富和声誉的机会。Web 3.0 会从哪里开始呢? 事实上,已经有了 Web 3.0,只不过还没有得到足够多的了解,那就是电子商务领域和在线游戏。不管是 B2C 还是 C2C,网民利用互联网提供的平台进行交易,在这个过程中,他们通过互联网进行劳动,并获得了财富。在线游戏通过积分的方式,角色扮演者通过攻城掠寨、不断地修炼、花费大量的时间,他们在那里可以获得声誉和财富,而这种财富通过一定的方式可以在现实中兑换,正所谓人生如同一场游戏,互联网会让人们的生活变得更像游戏一样。当前的论坛也提供积分,但由于缺乏个性,这个不会成为未来的主流,最有代表性的 Blog,却在积分方面做得很少,人们的劳动价值没有得到体现,正因为如此,好的博主将另起炉灶,以便得到更多,这是在追求一种更加均衡的分配方式。当这个 Web 2.0 的参与者有足够的力量和筹码的时候,他们就会要求一种对其更加公平合理的分配方式。在博主个人力量不够的时候,可以通过联合的方式来争取自己的利益。

2. Web 3.0 介绍

由上所述,可以得知 Web 3.0 是目前所能想象出来的未来互联网应用的框架。网站内信息可以直接和其他网站信息进行交互,能通过第三方信息平台同时对多家站点信息进行整合;用户在互联网上拥有自己的数据,并能在不同的网站上使用;完全基于 Web 访问,用浏览器即可实现复杂的系统程序才具有的功能。可以说 Web 3.0 是三广+三跨(广域的、

广语的、广博的，跨区域、跨语种、跨行业）。Web 3.0 阶段将是一个个性智能化的时代，无论使用者使用何种终端，只要打开浏览器就能进入自己的世界，就能看到自己所关心的内容，而不再需要盲目地搜索。世界知名技术未来预言家 Nova Spivack 认为 Web 3.0 是延伸至当前各大技术潮流迈向新的成熟阶段的具体体现，包括无处不在的互联网络、云计算、开放软件平台和数据、语义网技术、智能网络和智能应用程序。

3. Web 3.0 的特征

Web 3.0 将应用 Mashup 技术对用户生成的内容信息进行整合，使得内容信息的特征性更加明显，便于检索。将精确地阐明信息内容特征的标签进行整合，提高信息描述的精确度，从而便于互联网用户的搜索与整理。同时，对于 UGC（user generated content，用户原创内容）的筛选性过滤也将成为 Web 3.0 不同于 Web 2.0 的主要特征之一。对于互联网用户的发布权限经过长期的认证，对其发布的信息做不同可信度的分离，可信度高的信息将会被推到互联网信息检索的首项，同时提供信息的互联网用户的可信度也会得到相应的提高。

最后聚合技术的应用将在 Web 3.0 模式下发挥更大的作用，TAG（分类）/RSS（聚合内容）基础聚合设施，渐进式语义网的发展也将为 Web 3.0 构建完备的内容聚合与应用聚合平台。将传统意义的聚合技术和挖掘技术相结合，创造出更加个性化、搜索反应迅速、准确的“Web 挖掘个性化搜索引擎”。

（1）适合多种终端平台，实现信息服务的普适性

Web 3.0 的网络模式将实现不同终端的兼容，从 PC 互联网到 WAP 手机、PDA（personal digital assistant，掌上电脑）、机顶盒、专用终端，不只应用在互联网这一单一终端上。

现有的 Web 2.0 只能通过 PC 终端应用在互联网这一单一的平台上，面临现在层出不穷的新的移动终端的开发与应用都需要新的技术层面和理念层面的支持。而 Web 3.0 将打破这一僵局，使得各种终端的用户群体都可以享受到在互联网上冲浪的便捷。

实现融合网络的普适化、公用显示装置与个人智能终端的通用，同时加入 E-RAD 的应用与研发，使得嵌入式技术在 Web 3.0 模式下发挥更大的效力。

（2）良好的人性化用户体验以及基础性的个性化配置

Web 3.0 同样以人为本，将用户的偏好作为设计的主要考虑因素。Web 3.0 在对 UGC 筛选性过滤的基础上引入了偏好信息处理与个性化引擎技术，对用户的行为特征进行分析，既寻找可信度高的 UGC 发布源，同时对互联网用户的搜索习惯进行整理、挖掘，得出最佳的设计方案，帮助互联网用户快速、准确地搜索到自己感兴趣的信息内容，避免了搜索大量信息带来的疲劳。

个性化搜索引擎以有效的用户偏好信息处理为基础，对用户进行的各种操作以及用户提出的各种要求作为依据，来分析用户的偏好。再将通过偏好系统得出的结论归类到一起，在某个主题（如体育方面）上形成一种内容，并进行搜索的聚合、推送，从而更好地满足用户搜索、观看的需要。将这一技术引入广播电视中来，将会给传统电视带来巨大的影响。对于数字机顶盒的应用及 IPTV、WebTV 的推广提供了更好的聚合推送业务。

个性化引擎的建立是以偏好系统为基础的，偏好系统的建立要全面并且应与内容聚合相联系。有了一定的偏好分析，才能建立起完善的个性化引擎。

(3) 有效和有序的数字新技术

Web 3.0 将建立可信的 SNS(social networking services,社会网络服务系统),可管理的 VoIP(voice over Internet protocol,模拟声音信号数字化)与 IM(IMay 电子竞技俱乐部),可控的 Blog/Vlog/Wiki,实现数字通信与信息处理、网络与计算、媒体内容与业务智能、传播与管理、艺术与人文的有序及有效的结合和融会贯通。

Web 2.0 模式下的 SNS——网络社交平台,只是简单地将人与人通过互联网这一平台连接起来。通过互联网注册在 SNS 的平台上结交朋友,无法确保注册信息的可靠性和有效性,也并不是每一次交际圈的扩展都会带来相应的利益需求,这一过程进行下去的结果将会导致本身信息的外泄和零乱、不可靠信息的泛滥,颠覆了人们想利用互联网来扩展人际交往的初衷。这一问题在 Web 3.0 模式下,将通过对用户的真实信息的核查与认证的方式来解决。高可信度的信息发布源为以后交际圈的扩展提供了可靠的保障,与此同时,人们在交际的同时,也可以更迅速地找到自己需要的人才,并且可以完全信任这些可信度高的用户提供的信息,利用这些可以进一步扩展对自己有利的交际圈。

Web 3.0 模式下可管理的 VoIP 与 IM,同样为互联网用户的使用提供了方便快捷的服务方式。可信度越高、信用度越好的用户发布的信息越会被自动置顶,既提高了信息源发布者的可信度,同时使得这些有用、真实的信息更快地出现在用户的面前,从而发挥了信息的最大效力,提高了信息的使用率、降低了查找信息的时间损耗。

Web 3.0 模式下可控的 Blog/Vlog/Wiki,同样也是为了提高消息的利用率与查找信息的便捷性而产生的。这些原本在 Web 2.0 模式下允许用户随意发布的 Blog/Vlog/Wiki 会导致网络上堆积大量杂乱无章的信息,为用户的搜索带来了极大的不便。由此,Web 3.0 提出了“可控”这一概念,使得信息的发布与使用连接起来。如果想搜索高可信度的信息,可以单击可信度高的用户撰写的 Blog/Vlog/Wiki,实现可信内容与用户访问的对接。

4. Web 3.0 的前景

Web 3.0 化整为零,用户可以根据自己的喜好设计并建立属于自己的网页。Web 3.0 可以通过网页的剪切及粘贴功能,将自己喜欢的页面剪切并整理在一起,删除无用信息,而且最为重要的一点是所剪切的页面与主网页上的相关信息同步更新,不存在信息的滞后性,大大提高了阅读效率。Web 3.0 通过网页和相关组件的穿插,可以为使用者提供更为有效的信息资源,实现数字通信与信息处理、即时信息、交友娱乐、传播与管理的有序有效的结合。目前已知的相关企业有:百度空间、阔地网、天盟网、新浪博客、Google 等。

1.2.2　Web 新技术的应用

当前,以 HTML 5 为代表的新一代 Web 技术正处于技术发展的初期。由于众多科技巨头的鼎力支持,以及其自身强大的技术潜力和开放性,新一代 Web 技术已经得到了业界的高度关注,在技术发展和应用方面都进展顺利,被认为在未来有能力成为主导移动互联网产业发展的关键技术。可以预计,伴随着技术功能的不断完善以及标准化工作的顺利推进,新一代技术将在未来迎来高速发展。

当前全球主流浏览器厂商微软、谷歌、苹果、Mozilla 等企业都对以 HTML 5 为代表的新一代 Web 技术的发展给予了全面支持。新一代 Web 技术已应用于移动终端、桌面、电视等多个平台,初步实现了跨平台的快速发展。

1. 移动终端

在移动终端市场，由于苹果、谷歌等公司先后放弃了 Flash 技术转而支持 HTML 5 技术，推动了新一代 Web 技术在该领域的发展。而新一代 Web 技术的快速加载、本地/离线存储和地理位置获取功能也使得相关应用充分地发挥了移动设备的便捷特性，受到了开发者的青睐，因此未来的发展前景十分广阔。

2. 桌面应用

谷歌的 Chrome 浏览器、苹果的 Safari 浏览器和 Opera 浏览器一直对 HTML 5 有非常出色的支持。例如谷歌联合加拿大著名独立摇滚乐团共同推出了一个 HTML 5 互动电影 The Wilderness Downtown，页面上的动画效果皆由 HTML 5 技术制作，效果非常令人震撼。微软的 IE 10 浏览器对 HTML 5 的支持也表现良好，这无疑加速了 HTML 5 的发展。

3. 互联网电视

GoogleTV 以及 AppleTV 都已支持 HTML 5。互联网电视领域拥有大量的设备终端，能连接计算机的智能电视的数量正在不断增长。

4. 跨平台的 Web 应用

Mozilla 将推出跨平台 HTML 5 应用商店，它利用 HTML 5 的开源性，构建了一个支持 PC、手机和平板电脑的平台，用户不管使用何种设备或操作系统，都可以下载和安装自己喜欢的应用，让用户不再局限于一个特定的操作系统上，用户只需购买一次应用即可在任何启用 HTML 5 的设备上使用。

5. 网络游戏

HTML 5 网络游戏最大的优势就是平台的兼容性，能够同时支持 Android、iPhone 和 Windows Phone。腾讯旗下的手机 QQ 游戏大厅中的欢乐斗地主、QQ 斗地主等，是国内最早的 HTML 5 游戏开放平台之一。

1.3 Web 前端开发

1.3.1 什么是 Web 前端开发技术

Web 前端技术包括三个主要的因素：HTML、CSS、JavaScript。这三者之间虽然相互存在一定的关联性，但是在实际的运用过程中都具有自身的特点，对代码质量的要求也存在一定的差异性。HTML 技术可以对超文本的结构进行一定的探索，使得超文本语言在结构上更加完整，将一些过时的标记及时取消，内容和形式分离，进行技术结构的改进之后，生成的网页更加易于管理，可提升用户的体验。CSS，即层叠样式表技术，可用于增强网页样式的控制，为相关的信息和网页的分析提供一定的允许条件，是一种标记性的语言。这项技术的发展可以促进网页浏览速度的提升，对模块的维护和改善也具有一定的促进作用。JavaScript 技术出现之后，信息和用户之间的关系不仅仅是显示和浏览的关系，同时可以将实时、动态的数据进行一定的表达。JavaScript 可以和 HTML 技术进行结合，通过其在文件中的嵌入，将不需要进行整理的技术进行一定的响应，使得网页可以更好地与客户进行分

析，而不需要经过 Web 技术与客户进行交流，从而减少了服务器的压力。

随着 RIA（rich Internet applications，丰富互联网程序）的流行和普及，Flash/Flex、Silverlight、XML 和服务器端语言也成了前端开发的重要语言。随着时代的发展，前端开发技术的三要素也演变成为现今的 HTML 5、CSS 3 和 jQuery。

1.3.2　Web 前端开发工程师

Web 前端开发工程师是一个很新的职业，在国内乃至国际上真正开始受到重视的时间不超过 10 年。Web 前端开发是从网页制作演变而来的，名称上有很明显的时代特征。在互联网的演化进程中，网页制作是 Web 1.0 时代的产物，那时网站的主要内容都是静态的，用户使用网站的行为也以浏览为主。

2005 年以后，互联网进入 Web 2.0 时代，各种类似桌面软件的 Web 应用大量涌现，网站的前端由此发生了翻天覆地的变化。网页不再只是承载单一的文字和图片，各种丰富媒体让网页的内容更加生动，网页上软件化的交互形式为用户提供了更好的使用体验，这些都是基于前端技术实现的。

随着 Web 2.0 概念的普及和 W3C 组织的推广，网站重构的影响力正以惊人的速度增长。XHTML＋CSS 布局、DHTML 和 ajax 像一阵旋风，铺天盖地席卷而来，包括新浪、搜狐、网易、腾讯、阿里等在内的各种规模的 IT 企业都对自己的网站进行了重构。前端开发的入门门槛其实非常低，与服务器端语言先慢后快的学习曲线相比，前端开发的学习曲线是先快后慢。所以，对于从事 IT 工作的人来说，前端开发是个不错的切入点。

1. Web 前端开发工程师需求现状

上海互联网紧缺人才报告发布 Web 前端开发工程师最紧俏，人民网上海 2015 年 11 月 13 日电，《上海互联网行业人才紧缺指数（TSI）报告》显示，Web 前端开发工程师已成为上海互联网行业中最紧俏的职位。根据此次报告，2016 年第三季度，互联网在上海全行业中成为人才需求最为紧迫的行业；而在当地互联网各种紧缺职位中，各类研发技术类岗位稳居前列。

人才紧缺指数（talent shortage index，TSI）＝需求岗位数/求职人数。课工场中关村实训基地 TSI 大于 1，表示人才供不应求；小于 1，表示人才供大于求。如果 TSI 上升，表示人才紧缺程度加剧。猎聘网的报告指出，上海互联网 TSI 高达 2.49，在上海所有行业中排名第一，并与其他行业拉开明显的差距。

上海互联网行业自 2014 年第三季度以来，始终保持着较高的 TSI，互联网人才严重供不应求。到了 2015 年三季度，上海互联网 TSI 达到历史新高，高达 2.49。

在上海互联网行业的细分职能中，排名前十的绝大多数为技术性岗位，其中 Web 前端开发工程师最为紧俏，其 TSI 为 13.03。TSI 仅次于 Web 前端开发工程师的是软件工程师，其 TSI 为 6.02，环比增长 30.13％。

2. Web 前端开发工程师的职业要求

Web 前端开发工程师既要与上游的交互设计师、视觉设计师和产品经理沟通，又要与下游的服务器端工程师沟通，需要掌握的技能非常多。这就从知识的广度上对 Web 前端开

发工程师提出了要求。如果要精通于前端开发这一行，也许要先精通十行。然而，全才总是少有的。所以，对于不太重要的知识，我们只需要“通”即可。但“通”到什么程度才算够用呢？对于很多初级前端开发工程师来说，这个问题非常令人迷惑。

一位好的 Web 前端开发工程师在知识体系上既要有广度，又要有深度，所以很多大公司即使出高薪也很难招聘到理想的前端开发工程师。现在说的重点不在于讲解技术，而是更侧重于对技巧的讲解。技术非黑即白，只有对和错，而技巧则见仁见智。以前会 Photoshop 和 Dreamweaver 就可以制作网页，现在只掌握这些已经远远不够了。无论是开发难度上还是开发方式上，现在的网页制作都更接近传统的网站后台开发，所以现在不再叫网页制作，而是叫 Web 前端开发。Web 前端开发在产品开发环节中的作用变得越来越重要，而且需要专业的前端工程师才能做好，这方面的专业人才近两年来备受青睐。Web 前端开发是一项很特殊的工作，涵盖的知识面非常广，既有具体的技术，又有抽象的理念。简单地说，它的主要职能就是把网站的界面更好地呈现给用户。

所以一名优秀的前端开发工程师，不单单需要掌握前端必需的各种技术，同时还要掌握其他技术，需要掌握一点后台的知识，同时也要对网站构架有一定的了解，同时还要掌握一定的 SEO 网站优化技术，这样才可以称之为一个“优秀的 Web 前端开发工程师”。除了技术以外，还需要一定的时间来沉淀自己。一名资深的优秀 Web 前端开发工程师，是每个大型企业都渴望的人才。业内人士表示，宁可高薪招人，许多企业也不愿自己培养相关的技术人才。

如何才能做得更好呢？

第一，必须掌握基本的 Web 前端开发技术，其中包括 CSS、HTML、SEO、DOM、BOM、ajax、JavaScript 等，在掌握这些技术的同时，还要清楚地了解它们在不同浏览器上的兼容情况、渲染原理和存在的 Bug。

第二，在一名合格的前端工程师的知识结构中，网站性能优化、SEO 和服务器端的基础知识也是必须要掌握的。

第三，必须学会运用各种工具进行辅助开发。

第四，除了要掌握技术层面的知识，还要掌握理论层面的知识，包括代码的可维护性、组件的易用性、分层语义模板和浏览器分级支持，等等。

可见，看似简单的网页制作，如果要做得更好、更专业，真的并不简单。这就是前端开发的特点，也是让很多人困惑的原因。如此繁杂的知识体系让新手学习起来无从下手；对于老手来说，也时常不知道下一步该学什么。

代码质量是前端开发中应该重点考虑的问题之一。例如，实现一个网站界面可能会有无数种方案，但有些方案的维护成本会比较高，有些方案会存在性能问题，而有些方案则更易于维护，而且性能也比较好。这里的关键影响因素就是代码质量。CSS、HTML、JavaScript 这三种前端开发语言的特点是不同的，对代码质量的要求也不同，但它们之间又有着千丝万缕的联系。

1.4　Web 前端开发工具

优秀的工具可以使开发者的开发工作事半功倍，创建出高品质的 Web 应用程序。本节将介绍 8 款重要的 Web 开发工具，涵盖 CSS、JavaScript、图像优化及调试的 Web 开发工作，希望对开发者有所帮助。

1.4.1　CSS 工具

1. Layer Styles

如果不想去记新的 CSS 3 特性和前缀，Layer Styles 可帮上你的忙，该工具使用类似于 Photoshop 图层样式的界面，让你轻松配置阴影、背景、边框和边界半径。图 1-5 所示为 Layer Styles 站点。

网址：http://layerstyles.org/

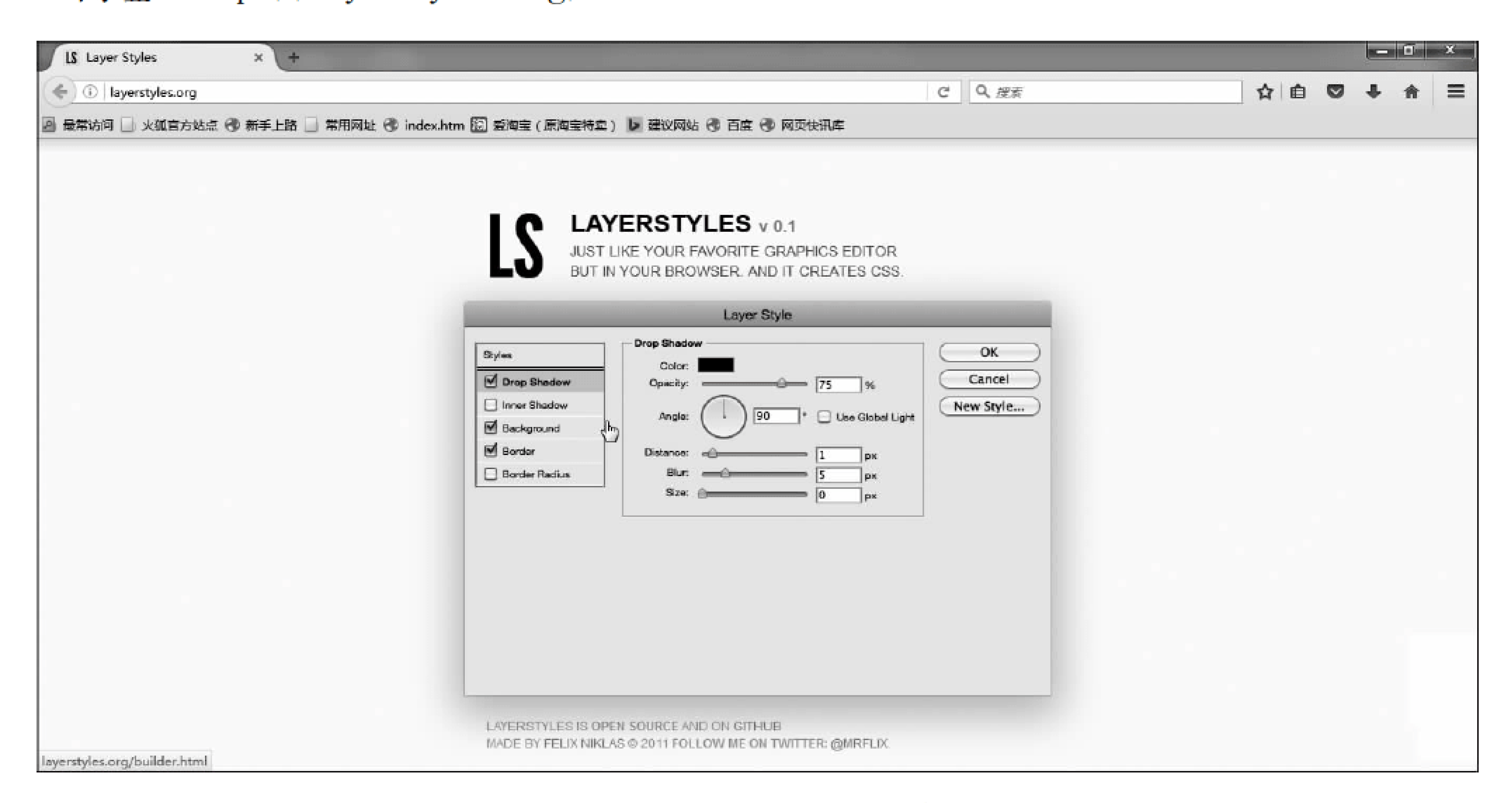


图 1-5　Layer Styles 站点

2. Bear CSS

该工具可以根据上传的 HTML 文档，采集其中所使用的 HTML 元素，来生成 CSS 模板。Bear CSS 站点如图 1-6 所示。

网址：http://bearcss.com/

1.4.2　JavaScript 工具

1. Bookmarkleter

该工具可以将你的 JavaScript 代码转换成一个小书签，帮助用户进行代码压缩和 URL 编码。Bookmarkleter 工具站点如图 1-7 所示。

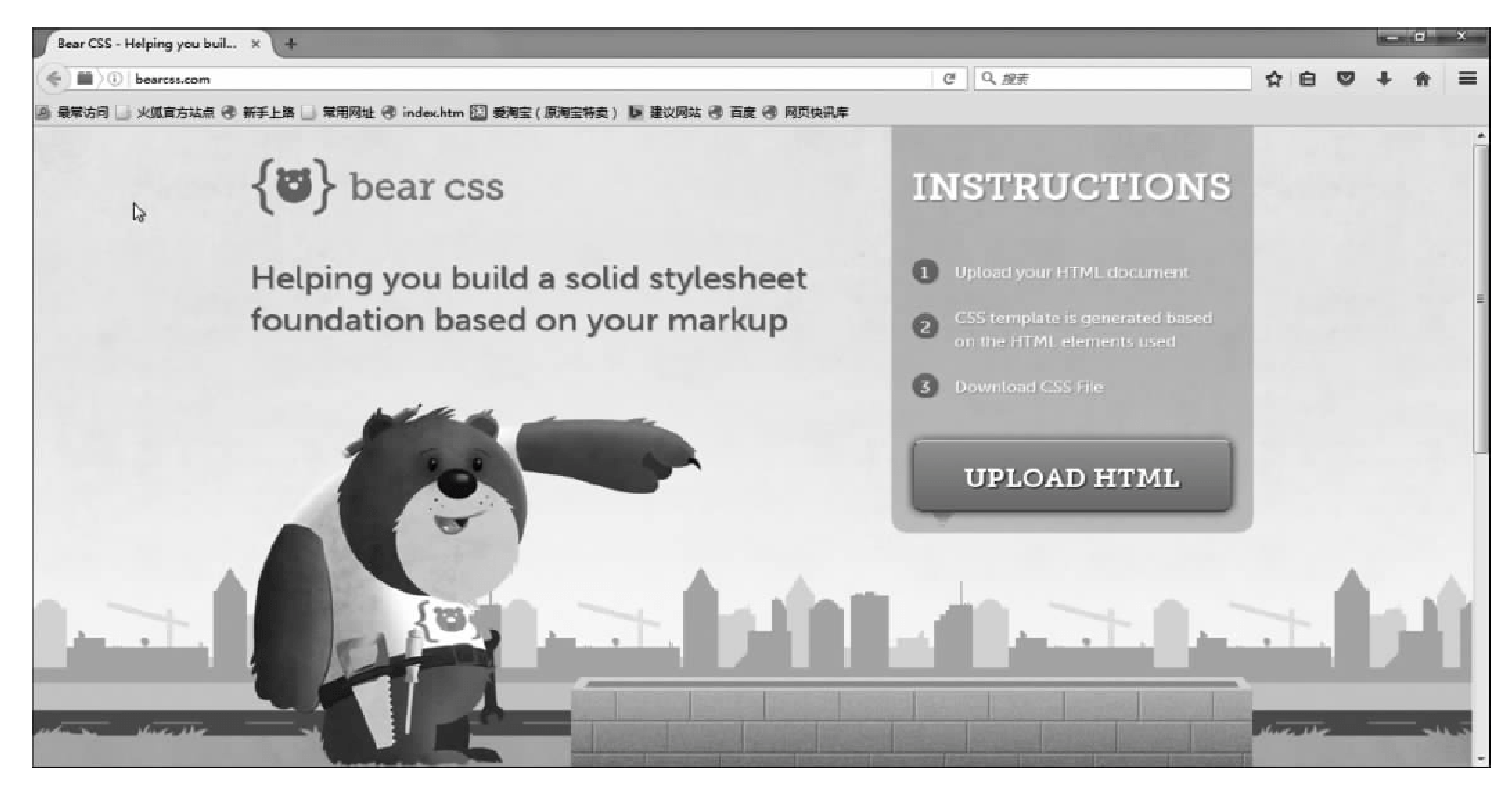


图 1-6　Bear CSS 站点

网址：https://chriszarate.github.io/bookmarkleter/



图 1-7　Bookmarkleter 工具站点

2．JSLint

这是一个 JavaScript 调试工具。可以查找代码中的潜在问题，并返回一条消息，告诉用户存在的问题以及出现的位置。JSLint 工具站点如图 1-8 所示。

网址：http://www.jslint.com/



图 1-8　JSLint 工具站点

1.4.3　图像优化工具

1. PunyPNG

该工具可以大大降低图像文件的大小，且没有任何质量损失。PunyPNG 工具站点如图 1-9 所示。

网址：http://conradchu.com/

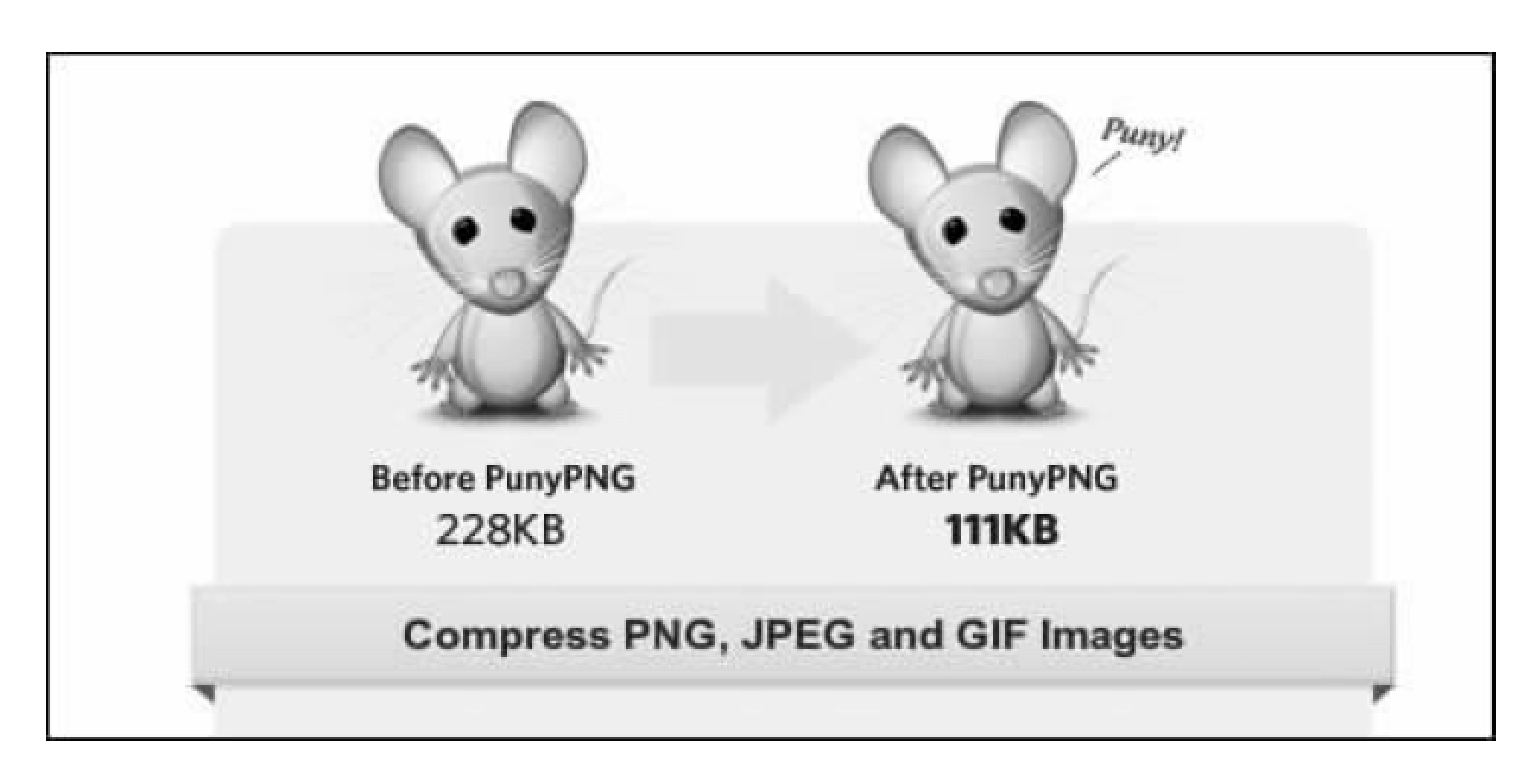


图 1-9　PunyPNG 工具站点

2. SpriteMe

该工具以一种不同的方式来创建 CSS Sprites。这是一个书签，可以检测用户网站中的图像，并相应地进行 CSS Sprites 的操作。SpriteMe 工具站点如图 1-10 所示。

网址：http://spriteme.org/

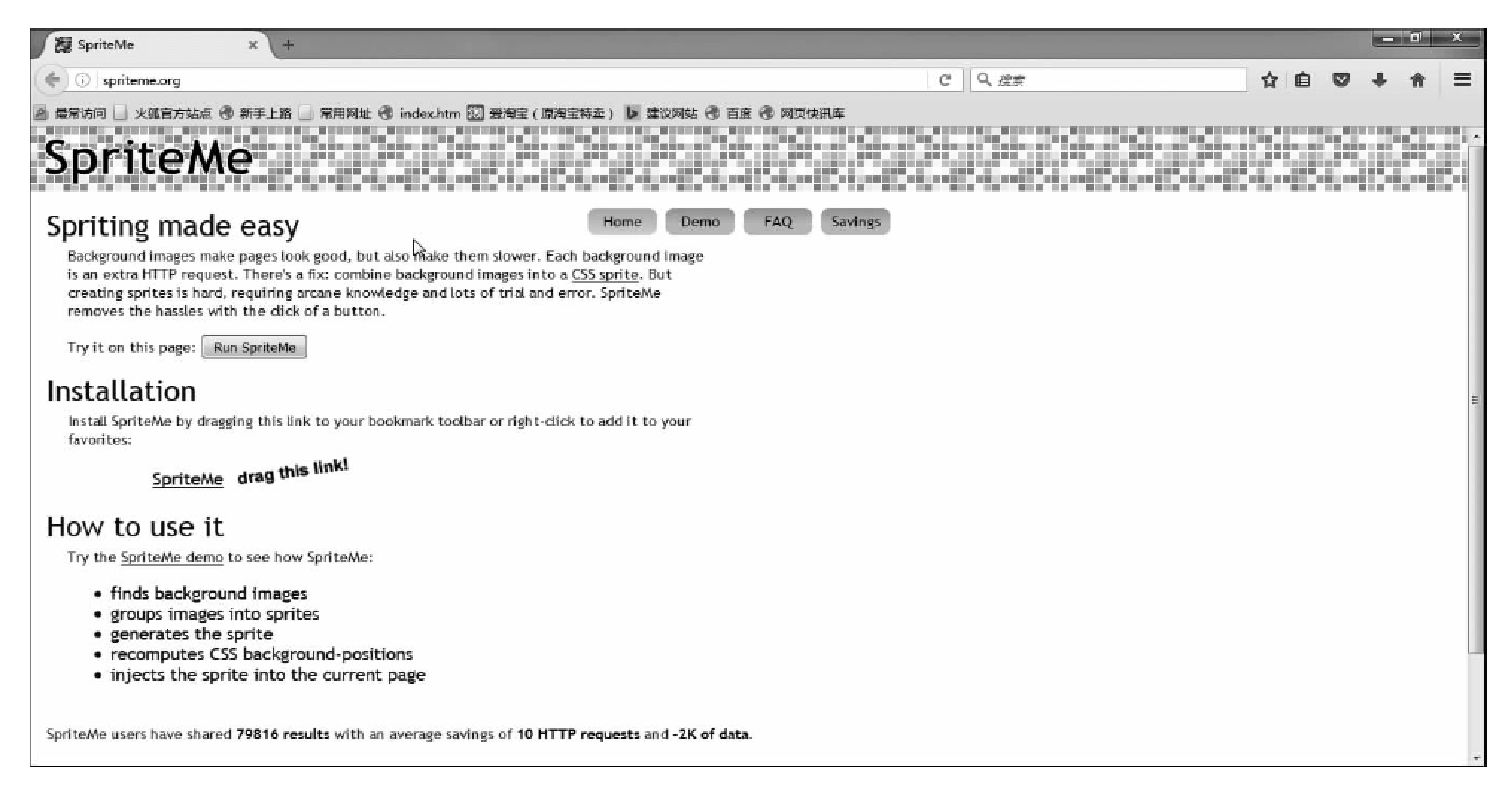


图 1-10　SpriteMe 工具站点

1.4.4　开发及调试工具

1. WebStore

WebStore 是 jetbrains 公司旗下的一款 JavaScript 开发工具，被广大中国 JavaScript 开发者誉为“Web 前端开发神器”“最强大的 HTML 5 编辑器”“最智能的 JavaScript IDE”等。与 IntelliJ IDEA 同源，继承了 IntelliJ IDEA 强大的 JavaScript 部分的功能。WebStore 工具站点如图 1-11 所示。

网址：http://www.webstore.com/

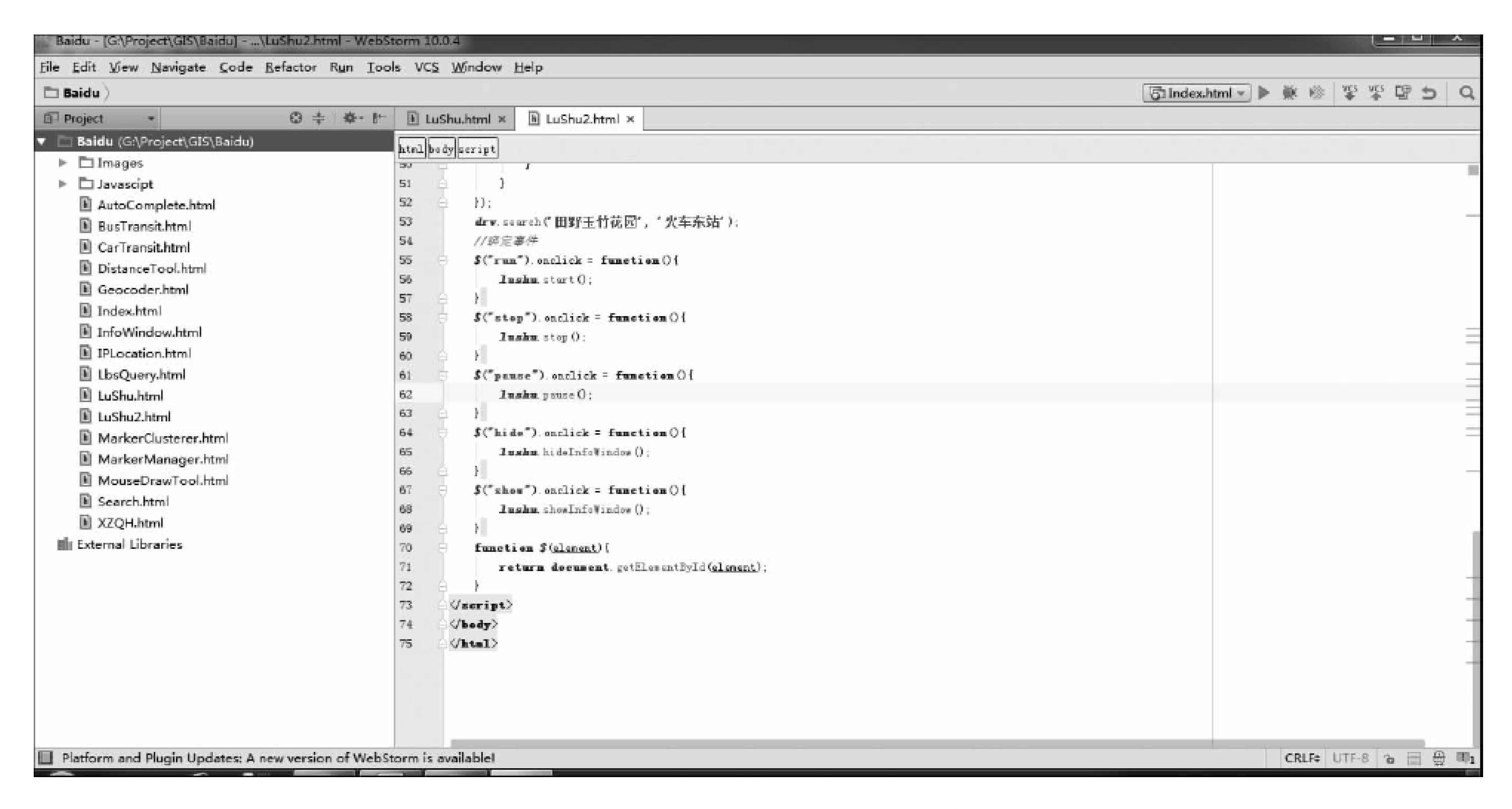


图 1-11　WebStore 工具站点

2．Fiddler

Fiddler 是一个 HTTP 协议调试代理工具，它能够记录并检查计算机和互联网之间的 HTTP 通信，设置断点，查看所有的"进出"Fiddler 的数据，包含 Cookie、HTML、JS、CSS 等文件，这些数据都可以让开发者在调试时随意修改。Fiddler 比其他的网络调试器更加简单，因为它不仅仅暴露 HTTP 通信，还提供了一个用户友好的格式。Fiddler 工具站点如图 1-12 所示。

网址：http://www.telerik.com/fiddler

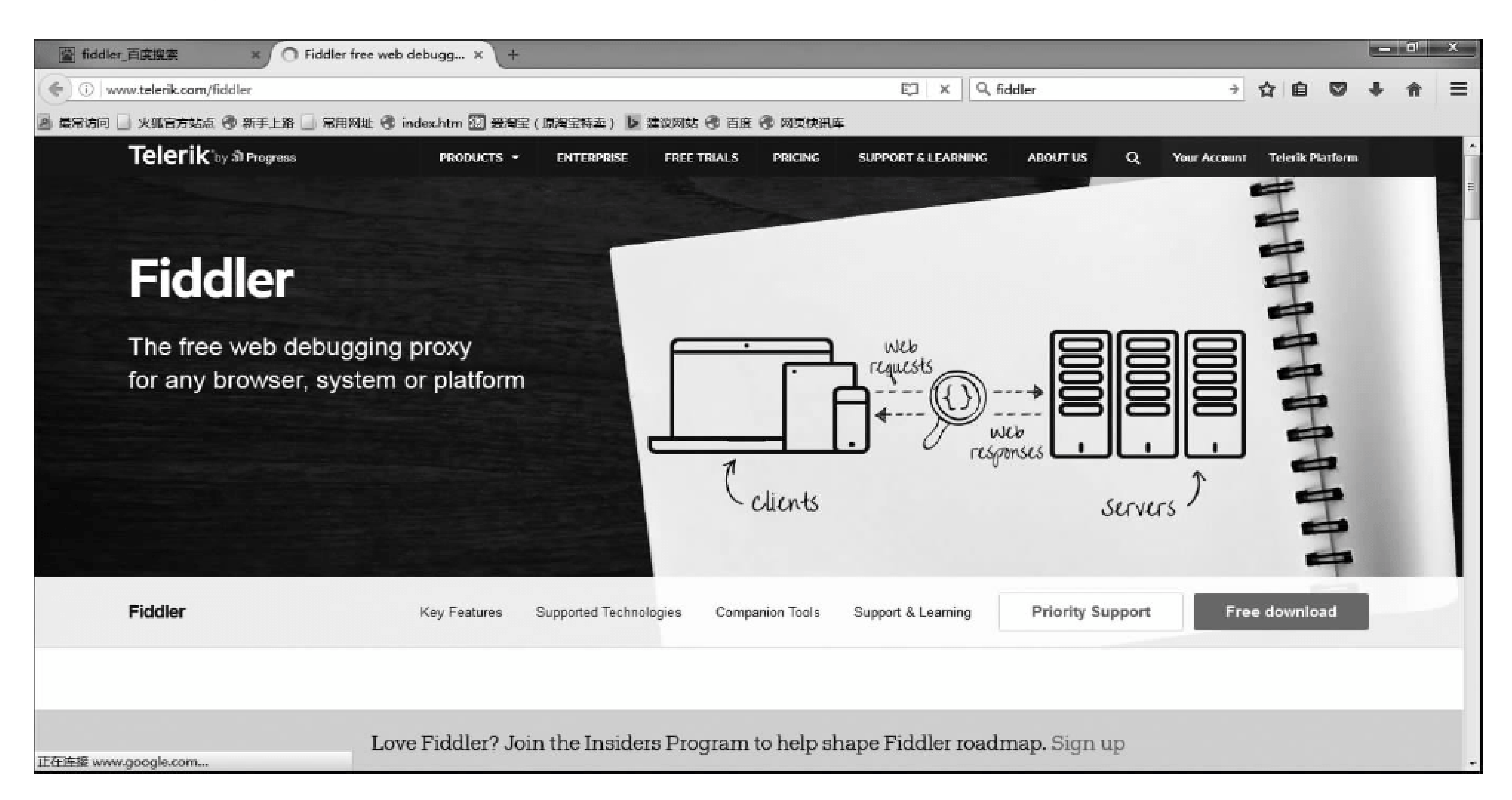


图 1-12　Fiddler 工具站点

本章小结

本章简述了 Web 的基础知识，包括 Web 的发展、Web 特点及架构、Web 3.0 及新技术的应用情况；本章还介绍了 Web 前端开发的相关知识，Web 前端开发工程师的职业要求；最后向读者推荐了 8 款 Web 前端开发工具，包括 CSS 工具、JavaScript 工具、图像优化工具和开发及调试工具。为读者学习后续的 Web 知识，进行 Web 设计、开发和应用提供了基础的概念和思路。

第 2 章　HTML 标记语言

2.1　HTML 基础

2.1.1　HTML 编写规范

HTML(HyperText Markup Language,超文本标记语言)是用来制作网页的一种规范和标准。HTML 文件对于很多平台具有很好的兼容性,可以通过网页浏览器在任何平台上运行并显示。

HTML 文件只是一个纯文本文件。所谓的“超文本”,是针对用 HTML 制作的网页页面来说的。通过 HTML 制作的网页页面内容可以包含图片、链接、音频、视频等非文本元素。HTML 文档其实是一种静态的网页文件,里面包含了 HTML 的指令代码,通过浏览器的编译和解释将页面中的内容按一定的排版位置显示出来。类似于传统的报纸杂志,然而这些纸质的媒体展示不出声音和视频,但网页可以。HTML 并不是一种程序设计语言,它只是一种排版网页中资料显示位置的标记结构语言,因此我们称它为超文本标记语言。

HTML 的指令代码是由标签组成的。标签是由“＜”和“＞”符号以及标签名和属性表示,其表示方式为“＜标签名 属性＞”。HTML 的标签只有两种类型,即单标签和双标签。

1. 单标签

单标签只有一对“＜”“＞”符号,表示形式为“＜标签名 属性＝参数＞”。最常见的单标签有强制换行＜BR＞、水平线＜HR＞、表单中的文本框＜INPUT＞等。在 HTML 中,单标签可以表示为＜标签名＞,但在 XHTML 中,表示为＜标签名 /＞,必须使用“/”符号正确地关闭标签。

2. 双标签

顾名思义,双标签则由两对“＜”“＞”符号组成,后一对的“＜”符号换成了“＜/”符号,表示形式为“＜标签名 属性＝参数＞...＜/标签名＞”,其中省略号部分为在网页页面中显示的内容,网页内容不能写在任何标签的“＜”“＞”符号内部。双标签中“＜/标签名＞”被称为结尾标签。结尾标签只能为“＜/标签名＞”的形式,里面不能包含任何属性。值得注意的是:在 HTML 文档中大多数采用的是双标签的形式。

需要说明的是,HTML 的标签名不区分大小写,即＜HTML＞和＜html＞表示效果是一样的。

2.1.2 HTML 基本结构

HTML 文档的基本结构由标签组成。

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>无标题文档</title>
</head>

<body>
</body>
</html>
```

HTML 文档的基本结构包括以下几种标签。

1. HTML 文件标签

<HTML>和</HTML>标签表示该文档为 HTML 网页文档。<HTML>标签放在文档的开头,</HTML>放在文档的末尾。网页的内容和其他标签都嵌套在这对标签之内。

2. HEAD 文件头标签

<HEAD>和</HEAD>标签表示文档的头部标签,用来描述文档的有关信息,如文档的标题、作者、采用的编码、关键词等。文档头部标签中的内容不会显示在浏览器的显示区域,主要提供给搜索引擎收录时使用。

(1) title 标签

title 标签是一种双标签。<TITLE>和</TITLE>标签之间的内容表示该文档的标题,显示在浏览器的标题栏中。title 标签只能用于 HTML 文档的头部<HEAD>标签中,且一个文档中只能有一个标题。

(2) meta 标签

<meta>标签是一种单标签。它提供有关页面的元信息(meta-information),比如针对搜索引擎和更新频度的描述和关键词。<meta>标签位于文档的头部标签中。由于是单标签,不包含任何内容,<meta>标签的属性是定义与文档相关联的名称/值对。

<meta>标签常用的属性有 content、http-equiv、name、scheme、charset,属性的值与描述请参考表 2-1。

表 2-1 <meta>标签的属性

属 性	值	描 述
content	some_text	定义与 http-equiv 或 name 属性相关的元信息
http-equiv	content-type	把 content 属性关联到 http 头部
	expires	
	refresh	
	set-cookie	

续表

属　性	值	描　　述
name	author	把 content 属性关联到一个名称
	description	
	keywords	
	generator	
	revised	
	others	
scheme	some_text	定义用于翻译 content 属性值的格式
charset	gb2312	定义文档使用的编码
	utf-8	

3. BODY 文件主体标签

＜body＞和＜/body＞标签为 HTML 文档的主体标签。网页中需要显示的所有内容都必须包含在这对标签之间。

2.1.3　查看 HTML 文件

HTML 文件可以直接通过浏览器解释展示出网页的内容。查看 HTML 文件的代码有多种方式，常用的有使用编辑工具查看和直接通过在浏览器的显示区域右击并选择“查看页面源代码”命令来进行查看。在网页的调试过程中，我们还常通过浏览器的开发调试(快捷键为 F12)查看源代码。

2.2　HTML 标签

2.2.1　文字与段落

文本在网页中的应用是很重要的。文本是网页中最基本的元素之一，也是浏览者通过网络获取信息的一种最主要的方式之一。网页中常见的文本标签有标题标签、段落标签和其他的一些为了展示特殊文本格式的常用标签。

1. 标题文字标签

网页的内容页中通常用一篇文章来描述信息，该信息一定存在标题，有可能还有多个标题。因此网页中提供了标题标签来表达网页中的标题。所谓标题文字就是以某种固定的字号显示的文字。标题文字包含 6 种层级，分别用标签 h1～h6 来描述，从 1 级到 6 级依次减小。

【例 2-1】　标题文字。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>标题文字</title>
</head>

<body>
    <h1>1 级标题文字</h1>
    <h2>1 级标题文字</h2>
    <h3>1 级标题文字</h3>
    <h4>1 级标题文字</h4>
    <h5>1 级标题文字</h5>
    <h6>1 级标题文字</h6>
</body>
</html>
```

程序的运行结果如图 2-1 所示。



图 2-1　标题文字

2. 段落文字标签

段落文字在页面内容中也是经常用到的。在 HTML 页面中，段落是通过<p>标签来表示的。段落标签是双标签，在使用时最好采用成对的<p>标签来包含段落。

【例 2-2】 段落文字。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>段落文字</title>
```

```
</head>

<body>
    <p>段落文字在页面内容中也是最常用到的。在 HTML 页面中，段落是通过 p 标签来表示的。
    段落标签是双标签，在使用时最好采用成对的 p 标签来包含段落。</p>
</body>
</html>
```

程序的运行结果如图 2-2 所示。

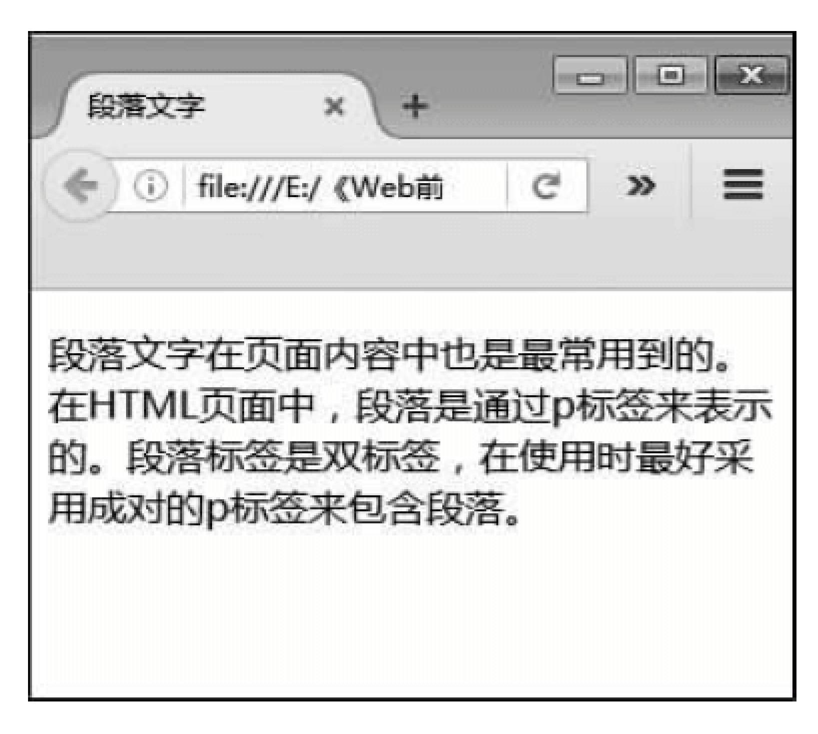


图 2-2　段落文字

3. 其他常用文字标签

除标题标签和段落标签外，页面内容还有很多格式需要展现。因此，HTML 还提供了一些其他的常用文字标签，如表 2-2 所示。

表 2-2　其他常用的文字标签

标　签	语　　法	描　　述
b	…	字体加粗
strong	…	字体加强
em	…	文字强调
i	<i>…</i>	斜体
sub	_…	下标
sup	[…]	上标
del	~~…~~	删除线
span	…	其他任何文字标签都不适用，但又要在使用特殊文字格式时用到

由于以上除 span 标签外的其他文字标签所展现出来的文字格式都可以通过样式设置来完成，因此，在 HTML 5 中，这些标签都被取缔，而通过样式设置来实现特殊的文字格式。

4. 转义字符

HTML 页面所使用的标签是一些带有“<”“>”尖括号的普通字符，如果页面内容中需要显示出尖括号或带有尖括号的字符，浏览器则会将这些带有尖括号的符号识别为 HTML 的标签，不会显示在页面内容中。因此，HTML 提供了一些转义字符来表达这些特殊字符，

见表 2-3。

表 2-3　转义字符

特殊字符	转义码	特殊字符	转义码
空格		& 符号	&
双引号(" ")	"	￥符号	¥
单引号(' ')	'	乘号(×)	×
小于(<)	<	除号(÷)	÷
大于(>)	>	已注册商标(　)	®
版权号(　)	©		

在 HTML 中，转义字符的编码是以“&”符号开始，以“;”符号结束，中间为相关转义字符的编码。“&”和“;”符号是不可省略的。

页面中的空格符号也是需要转义字符来实现的。文字换行也需要换行标签
，直接通过文字的换行排版是行不通的。

2.2.2　图片与超链接

除文字外，图片也是网页中最重要的元素之一。合理使用图片可以使网页增添不少色彩。网页支持多种图片格式，可以是 GIF、JPEG、PNG、BMP 等格式。其中 GIF 和 JPEG 格式是网页中最常用的。近年来具有不失真且支持透明的 PNG 格式图片也经常用在网页中，特别是移动端的网页应用最为广泛。

1. 插入图片

网页中使用图片有两种形式，一种是在网页中直接插入图片；另一种是将图片作为背景样式设置在网页中。网页中插入图片使用<img>标签(见表 2-4)。<img>标签是单标签，仅使用该标签并不能将图片插入网页中，必须要有图片来源的标签属性 src。

表 2-4　<img>标签的属性及描述

属　性	值	描　述
src	URL	将要插入图片的 URL
width	pixels	设置图片的宽度
height	pixels	设置图片的高度
alt	text	图片无法正常显示时代替图片显示的文字
title	text	鼠标放图片上时显示的有关文本描述

src 属性用于指定图片源文件的路径，它是<img>标签必不可少的属性，使图片显示在网页中。

【例 2-3】　插入图片。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>插入图片</title>
```

```
</head>

<body>
    <img src="images/hua.jpg" width="240" height="170" title="鲜花" alt="美丽的
    花朵" />
</body>
</html>
```

程序的运行结果如图 2-3 所示。



图 2-3　插入图片(正常显示)

<img>标签中的 src 属性指定了图片源的路径,width 属性指定图片在网页中的显示宽度,height 属性指定图片在网页中的显示高度,title 属性是鼠标光标移到网页中显示图片上时所展示出来的文本描述,alt 属性是图片在浏览器中无法正常显示时代替图片所显示的文本信息,如图 2-4 所示。



图 2-4　插入图片(不能正常显示图片)

2. 超链接

Web 上的网页都是互相链接的，通过点击一个页面上的超链接可以进入其他页面。

网页中超链接是通过<a>标签（见表 2-5）完成的。<a>标签是双标签。在成对的<a>标签之间的内容即为网页上所显示的可单击的超链接对象，该对象可以是文字，也可以是图片。与<img>标签类似，超链接也不仅仅是通过<a>标签实现的，必须要有链接的目标属性 href。

表 2-5　<a>标签的属性及描述

属　性	值	描　　述
href	URL	超链接目标文件的 URL
target	_blank	链接目标的窗口打开方式
	_parent	
	_self	
	_top	

href 属性指定超链接将要链接到的目标文件的路径，target 属性指定链接目标的窗口打开方式。在使用<a>标签时，target 属性可以省略，省略时链接将在当前窗口打开。若要在新的浏览器窗口中打开链接，target 属性值设置为：target＝"_blank"。由于 HTML 5 已经取消了框架集（frameset），所以 target 的其他三个属性值（target＝"_parent"、target＝"_self"、target＝"_top"）也基本没有用。

【例 2-4】 文字和图片超链接。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>文字和图片超链接</title>
</head>

<body>
    <a href="例 2-2.html" target="_blank">文字链接</a>
    <a href="例 2-3.html"><img src="images/hua.jpg" width="240" height="170" />
    </a>
</body>
</html>
```

程序的运行结果如图 2-5 所示。

超链接<a>标签的 href 属性指向链接的目标文件，而链接的目标文件可以是各种类型的文件。超链接不仅可以链接到图片文件、声音文件、视频文件、Word 文档等，还可以链接到其他网站、FTP 服务器、电子邮件等。



图 2-5　文字和图片超链接

2.2.3　列表标签

列表标签也是网页中常用的标签。列表可以有序地对信息资源进行排版，使页面结构具有条理性。HTML 提供了无序列表、有序列表和定义列表三种列表标签。

1. 无序列表

无序列表就像是 Word 中的项目符号，列表项没有排列顺序，只有列表项目符号。无序列表用一对
标签表示，列表项用一对
标签表示。项目符号则用<ul>标签的 type 属性表示。

无序列表<ul>的 type 属性见表 2-6。

表 2-6　无序列表<ul>的 type 属性

属　性　值	描　述
disc(默认值)	●
square	■
circle	○

在一个无序列表中可以包含多个列表项，列表项的结尾标记
可以省略，但为了结构的完整性，最好不要省略。列表项中可以嵌套列表。

【例 2-5】　无序列表。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>无序列表</title>
</head>
<body>
    <ul>
        <li>一级标题</li>
        <li>一级标题</li>
```

```
        <li>一级标题
            <ul>
                <li>二级标题</li>
                <li>二级标题</li>
                <li>二级标题</li>
            </ul>
        </li>
        <li>一级标题</li>
        <li>一级标题</li>
        <li>一级标题</li>
    </ul>
</body>
</html>
```

程序的运行结果如图 2-6 所示。



图 2-6 无序列表

嵌套列表在未设置项目符号时，会默认以另一种符号显示。

2. 有序列表

有序列表类似于 Word 中的编号列表，列表项按照一定的顺序排列。有序列表采用一对
标签表示，列表项依然采用一对
标签表示，但列表项具有一定的顺序。列表项符号通过 type 属性设置，如表 2-7 所示。

表 2-7 有序列表<ol>的 type 属性

属性值	描 述	属性值	描 述
1	数字(默认)	I	大写罗马数字
A	大写字母	i	小写罗马数字
a	小写字母		

有序列表项中也可以嵌套列表，嵌套的列表可以是有序列表，也可以是无序列表。有序列表的列表符号从前到后依次排序，默认情况下都是从 1 开始。如果第一个列表项的项目符号不从 1 开始，则使用 start 属性设置。然而 start 属性的属性值只能是数字“1，2，3，…”。如果第一个列表项符号从“F”开始，并不是将 start 属性值设置为：start="F"，而是将 start

属性值设置为：start="6"。

【例 2-6】 有序列表。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>有序列表</title>
</head>
<body>
    <ol>
    <li>有序列表一级列表
        <ul>
            <li>有序列表二级列表</li>
            <li>有序列表二级列表</li>
            <li>有序列表二级列表</li>
        </ul>
    </li>
    <li>有序列表一级列表</li>
    <li>有序列表一级列表
        <ol type="A" start="6">
            <li>有序列表二级列表</li>
            <li>有序列表二级列表</li>
            <li>有序列表二级列表</li>
        </ol>
    </li>
    <li>有序列表一级列表</li>
    <li>有序列表一级列表</li>
    <li>有序列表一级列表</li>
    </ol>
</body>
</html>
```

程序的运行结果如图 2-7 所示。

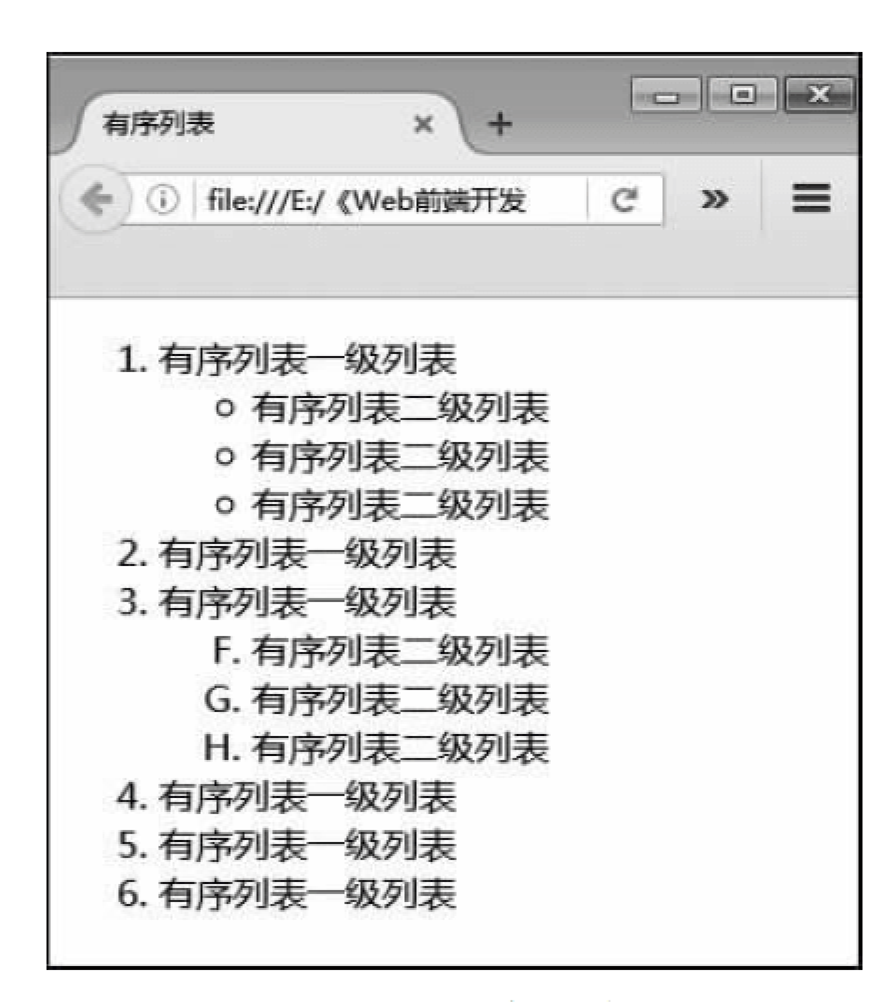


图 2-7 有序列表

3. 定义列表

定义列表用来展示术语及其解释。定义列表用一对<dl></dl>标签表示，列表项中的术语用<dt></dt>标签表示，列表项中的解释用<dd></dd>标签表示。术语和解释都各占一行，术语顶格显示，解释则换行缩进显示。

【例 2-7】 定义列表。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>定义列表</title>
</head>

<body>
    <dl>
        <dt>名词解释</dt>
        <dt>无序列表</dt>
        <dd>无序列表就像是 Word 中的项目符号,列表项没有排列顺序,只有列表项目符号。
        </dd>
        <dt>有序列表</dt>
        <dd>有序列表类似于 Word 中的编号列表,列表项按照一定的顺序排列。</dd>
        <dt>定义列表</dt>
        <dd>定义列表用来展示术语及其解释。</dd>
    </dl>
</body>
</html>
```

程序的运行结果如图 2-8 所示。



图 2-8 定义列表

定义列表中的列表项“术语”和“解释”并不一定要交替列示，可以连续出现多对<dt></dt>标签，也可以连续出现多对<dd></dd>标签，还可以交替出现。

2.2.4 表格标签

在 HTML 中创建表格的语法是完全按照表格的自然构成来组织的。

1. 创建表格

在 Word 或者 Excel 中的表格具有行和列，在网页中表格同样具有行和列。在 HTML 文档中创建表格时，首先需要确定创建几行几列的表格，然后才开始创建。因此，表格由三个基本元素组成：table 元素、tr 元素、td 元素。

(1) table 元素：用来定义表格，整个表格包含在<table>和</table>标签中。

(2) tr 元素：用来定义表格中的行。一对<tr>和</tr>标签表示表格中的一行。它也是单元格容器，一行中可以包含若干个单元格。

(3) td 元素：表格的列标记，也是表格的单元格，包含在表格的行标记<tr>中。每个单元格用一对<td>和</td>标签表示。

(4) th 元素：有时候会看到表格中存在 th 元素，其实它跟 td 元素一样也可以表示表格的单元格，所不同的是，它可以用来创建表格的头信息单元格，俗称表头元素，一般用在表格的第一行或者第一列。表头元素从样式上看，其实就是对单元格内的文字进行了加粗设置。因此，在使用表格时不常用表头元素，而直接采用单元格 td 元素，只需要设置样式来达到表头的显示效果。

(5) caption 元素：表格的标题标签。通过它可以对创建表格的目的和作用进行一个简单的说明。caption 元素内的内容即为该表格的标题。caption 元素只能被定义在 table 元素的开始标签之后，tr 元素之前，并且一个表格即一个 table 元素中仅能定义一个 caption 元素。

表格在网页中的结构表现与自然结构是相同的，具有行和列。因此表格的基本结构如下：

```
<table>
    <tr>
        <th> </th>
        <th> </th>
    </tr>
    <tr>
        <td> </td>
        <td> </td>
    </tr>
</table>
```

表格<table>标签中包含行<tr>标签，在行标签中包含列<td>标签。

2. 单元格的合并与拆分

(1) 横向合并单元格

横向合并单元格就是将一行中的几个单元格进行合并，合并的是单元格的列数，用 colspan 属性来进行设置，相应的这一行中的单元格个数对应减少。属性值是一个数字，表示合并的单元格的个数。

（2）纵向合并单元格

纵向上的单元格合并是将同一列中的几个单元格进行合并，合并的是这一列单元格的行数，用 rowspan 属性来进行设置，相应的其他行中的单元格个数对应减少。它的属性值也是一个数字，表示合并的单元格的个数。

在我们对单元格合并的实际应用中，并不一定只有横向的合并或者只有纵向的合并，有时候既有横向的合并，也有纵向的合并，即将相邻的多行多列单元格进行合并。合并后的单元格中既有 colspan 属性，也有 rowspan 属性。

（3）单元格的拆分

一个单元格可以拆分成多个单元格。单元格拆分时可以拆分成多行，也可以拆分成多列，但是一次拆分只能操作一个单元格，并且一次拆分要么拆分成多行，要么拆分成多列，不能一次操作既有多行又有多列的拆分。既有多行又有多列的拆分可以分多次进行。

单元格的拆分，其实质是将其他相关的单元格进行合并操作。如：将一行中的一个单元格拆分成两行的两个单元格，而这一行中其他的所有单元格，都在纵向上合并了两个单元格，分别增加了 rowspan＝"2"的属性，并且表格会增加一行，即多了一对＜tr＞标签，＜tr＞标签中只有一对＜td＞标签。

3. 表格的嵌套

表格可以用来布局网页，但是网页的栏目板块有时候是以不规则形式展现的。因此，要用标准的表格去布局不规则的网页栏目时，会采用一定的表格嵌套来完成。

表格的嵌套就是在建立的表格中的某一个单元格当中，再创建一个表格。

```
<table width="600" border="1">
    <tr>
        <td width="150" height="200"> </td>
        <td width="150" height="200"> </td>
        <td width="150" height="200"> </td>
        <td width="150" height="200"> </td>
    </tr>
    <tr>
        <td height="100" colspan="4"><table width="100%" border="1">
            <tr>
                <td width="200" height="200"> </td>
                <td width="200" height="200"> </td>
                <td width="200" height="200"> </td>
            </tr>
        </td>
    </tr>
</table>
```

表格的嵌套常用在网页的布局中。

2.2.5　表单标签

HTML 表单通常在网页中表现为注册、登录页面，调查信息表、订单页面以及一些搜索界面等。这些页面主要是用来搜集用户信息，并且将这些收集来的信息发送到服务器端进

行处理。因此，表单是客户端与服务器端传递数据的桥梁，也是用户与服务器之间实现相互交互的最主要的方式。

1. 创建表单

网页中的表单用<form></form>标签进行创建。表单标签是双标签，用一对标签分别定义表单的开始位置和结束位置，在标签对之间创建表单控件。在表单的开始标签<form>中，可以设置表单的基本属性，包括表单的名称 name 属性，处理表单数据的目标程序 action 属性以及传送数据的方法 method 属性等。表单<form>标签相当于是表单的容器，里面除了包含表单控件外，还可以包含其他的文本元素，如段落、列表等。表单标签中常用的属性如表 2-8 所示。

表 2-8　表单标签中常用的属性

属　性	描　　述
name	用来设置表单的名称
action	设定表单数据处理程序 URL 的地址
method	用来定义数据传递到服务器的方式

表单标签中的 method 属性如表 2-9 所示。

表 2-9　表单标签中的 method 属性

属性值	描　　述
get	将表单中的数据加在 action 指定的地址后面并传送到服务器中，即将表单控件中的输入数据按照 HTTP 传输协议中的 GET 传输方式传送到服务器端。这种方式传送的字段小，安全性低
post	将表单中所有控件的输入数据按照 HTTP 传输协议中的 POST 传输方式传送到服务器中。这种传输方式传送的字段大，安全性高
put	将输入数据按照 HTTP 传输协议中的 PUT 传输方式传送到服务器中
delete	将输入数据按照 HTTP 传输协议中的 DELETE 传输方式传送到服务器中

2. 表单控件

用户与表单交互是通过表单的控件进行的。表单控件通过 name 属性进行标识，通过 value 属性值获取输入数据。表单的“提交”是通过表单的“提交”按钮完成的。

表单控件中，input 元素可以定义表单中的大部分的控件，控件的类型由 type 属性(见表 2-10)决定，不同的值对应不同类型的表单控件。

表 2-10　input 元素的 type 属性

属　性	控件的类型说明
text(默认)	表示单行输入文本框
password	表示密码框，输入的数据用星号显示
radio	表示单选框
checkbox	表示复选框
file	表示文件域，由一个单行文本框和一个“浏览”按钮组成

续表

属　性	控件的类型说明
submit	表示“提交”按钮，将表单数据发送到服务器
reset	表示“重置”按钮，将重置表单中的数据，以便重新输入
button	表示普通按钮，应用 value 的属性值定义按钮上显示的文字
image	表示一个图像按钮
hidden	表示隐藏文本框

除了 type 属性，input 元素还有以下一些常用的属性。

(1) name 属性：为表单控件定义一个名称标识，这个名称将与控件的当前值组成“&名称=值”对，并一同随着表单数据进行提交。

(2) value 属性：用于指定初始值，即默认的显示值。当文本框中没有输入信息时，在网页中显示出来的是初始值。它是可选的，可以不设置。但是 value 属性非常重要，因为它的值将会被发送到服务器，即使是单选框或复选框，最好都设置好 value 属性，这样提交数据时也可以将单选按钮和复选框中用户所选择的信息提交出去。

(3) size 属性：设置表单控件的初始宽度，值是以字符的个数为单位。

(4) checked 属性：这个属性只针对单选按钮和复选框进行设置。它是一个逻辑值，指定单选按钮或复选框是否处于选中状态。当表单控件(单选按钮或复选框)中设置该属性时，表示该选择框被选中；没有设置则表示该选择框没有被选中。checked 属性只用于单选按钮和复选框，且只有一个属性值 checked。

(5) maxlength 属性：指定表单控件中可以输入的最大字符数，数值可以超过 size 属性设置的数值。该属性常用于单行输入文本框和密码框。如果控件中不设置该属性，表示该控件对输入的字符数没有限制。

(6) src 属性：只针对 type="image"的图像按钮，用来设置图像文件的路径。

还可以设置 readonly 属性，用于在文本框中显示文本，而不能输入数据。

2.2.6　注释标签

网页稍大一些，结构就会显得复杂。在 HTML 源代码中适当加入注释语句，能够使网页结构代码更清晰，更具有可读性，这也是设计者编写代码的一种好习惯，便于以后修改。另外，其他的设计者拿到具有注释的源代码也很容易上手。

所谓注释，是在 HTML 代码中插入用来解释或提示性的描述信息文本。注释语句只出现在代码中，浏览器不会对注释代码进行解释从而显示在浏览器的展示区域。

HTML 中的注释标签为“＜!--注释的内容--＞”，可以注释一行或多行代码，而且可以注释 HTML 代码。只要将要注释的代码放在“＜!--”和“--＞”之间，这些代码就不会在浏览器中显示出来。

【例 2-8】 注释标签。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.
org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>注释标签</title>
</head>

<body>
    <h3>以下是注释内容</h3>
    <!--注释内容
    <a href="例 2-2.html" target="_blank">文字链接</a>
    <a href="例 2-3.html"><img src="images/hua.jpg" width="240" height="170" />
    </a>
    -->
</body>
</html>
```

程序的运行结果如图 2-9 所示。



图 2-9　注释标签

源代码中被注释的代码不会在浏览器中显示。

2.3　XHTML 基础

2.3.1　XHTML 简介

XHTML(eXtensible HyperText Markup Language，可扩展的超文本标记语言)是一个基于 XML 的标记语言，是为取代 HTML 而存在的。XHTML 是一种过渡技术，XHTML 1.0 是一种在 HTML 4.0 基础上优化和改进的新语言，目的是基于 XML 应用。XHTML 是一种增强了的 HTML，XHTML 是更严谨、更纯净的 HTML 版本。它的可扩展性和灵活性将适应未来网络应用更多的需求。XHTML 1.0 Transitional 版本与 HTML 4.01 几乎是相同的。

2.3.2　XHTML 语法

XHTML 比 HTML 更简洁，语法也更严谨。本书所制作的 HTML 页面所使用的便是

XHTML 1.0 Transitional 版本。

XHTML 遵循以下几点语法规则。

(1) XHTML 标签的属性名称必须小写。

(2) XHTML 属性值必须加引号。

(3) XHTML 属性不能简写。如表单控件<input>标签的 checked 属性,不能写成<input checked>,必须要写成<input checked="checked">,这样才是正确的。

(4) 用 id 属性代替 name 属性。

(5) XHTML DTD 定义了强制使用的 HTML 元素。所有 XHTML 文档必须进行文件类型声明(doctype declaration)。在 XHTML 文档中必须存在 html、head、body 元素,而 title 元素必须位于 head 元素中。但文件类型声明并非 XHTML 文档自身的组成部分,它并不是 XHTML 元素,也没有关闭标签。

以下代码是 Dreamweaver 新建的一个 XHTML 文件结构。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>无标题文档</title>
</head>
<body>
</body>
</html>
```

在 XHTML 中,<html>标签内的 xmlns 属性是必需的。然而,即使当 XHTML 文档中没有这个属性时,w3.org 的验证工具也不会提示错误。这是因为 xmlns 是一个固定的值,即使没有把它包含在代码中,这个值也会被添加到<html>标签中。

2.3.3　XHTML 与 HTML 的区别

HTML 和 XHTML 的区别简单来说,可以认为 XHTML 是 XML 版本的 HTML,为了符合 XML 的要求,XHTML 语法上要求更严谨。

(1) XHTML 所有的标签都必须要有相应的结尾标签。在 HTML 中,
标签可以不用关闭,即不用
标签也是可以的,但在 XHTML 中,必须要有严谨的结构,所有标签必须关闭,否则是不合法的。例如单标签,在标签结束的右尖括号前也必须加上"/"来关闭。

(2) XHTML 的标签名是区分大小写的。

(3) 所有的 XHTML 标签都必须合理嵌套。因为 XHTML 要求有严谨的结构,因此所有的嵌套都必须按顺序,如<p></p>这样的嵌套是正确的,然而如果写成<p></p>,则是不正确的。合理的嵌套必须是严格对称的。

(4) 所有的属性必须用引号("")括起来。

(5) 所有<和 & 等特殊符号用转义码表示。

任何小于号(<)如果不是标签的一部分,都必须使用转义码"<";任何大于号(>)

如果不是标签的一部分，都必须使用转义码“>”，任何“与”号(&)如果不是实体的一部分的，都必须使用转义码“&”。

(6) XHTML 的所有属性不能简写。XHTML 规定所有属性都必须有一个值，没有值的就重复本身。例如，<input type="checkbox" checked="checked">。

(7) XHTML 不能在注释内容中使用“--”符号。“--”只能写在 XHTML 注释的开头和结尾，不能出现在内容的中间。例如下面的代码是无效的：“<!--这里是注释-----------这里是注释-->”。但用等号或者空格替换内部的虚线：“<!--这里是注释============这里是注释-->”，这样写就是合法的。

(8) XHTML 中插入的图片必须要有说明文字。每个图片标签都必须有 alt 属性来表述说明文字。比如：。为了兼容火狐和 IE 浏览器，对于图片标签，尽量采用 alt 和 title 双属性，单纯的 alt 属性在火狐下没有图片说明。

2.4 HTML 5

2.4.1 HTML 5 文档结构

HTML 从版本 1.0 到版本 5.0 发生了巨大的变化，也做出了很大的改进。尤其是 HTML 5 对多媒体的支持更加强大。HTML 5 新增了离线存储、新的文档对象模型、跨文档消息等功能，因此，HTML 5 的应用越来越广泛，更多地应用于移动端的网页。

HTML 5 的文档结构与 HTML 的文档结构类似，包括文档开始、结束的标签，文档的头部标签，文档的主体标签。HTML 5 文档的基本结构如下：

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>无标题文档</title>
</head>
<body>
</body>
</html>
```

从以上的结构代码中可以看出，HTML 5 的结构代码比以前的版本都简洁。结构代码中的标签都是前面介绍过的，这里就不赘述了。

2.4.2 HTML 5 新增的结构标签及属性

为了适应浏览器的多样性，HTML 5 中新增了一些结构标签和属性，以及取缔了一些没多大作用的标签和属性，下面详细进行介绍。

1. HTML 5 新增的标签

HTML 5 中新增的结构标签及描述如表 2-11 所示。

表 2-11　HTML 5 中新增的结构标签及描述

结构标签	描　　述
section	定义文档或应用程序中的一个区段，如章节、页眉、页脚等。可以配合标题标签使用，标识文档结构
article	定义文档中的一块独立内容，如博客或报纸中的一篇文章
header	定义页面中的一个内容区块或整个页面的标题
nav	定义页面中的导航
footer	定义页面中的一个内容区块的脚注，一般包含版权信息
aside	定义页面中的侧边栏目，一般是与 article 相关的内容
figure	定义文档主体流内容中的一个独立单元，类似于文章中的二级主体内容
dialog	定义对话，如交谈。标签里面可以使用定义列表项<dt>和<dd>标签
mark	行内元素，在视觉上呈现需要突出显示或高亮显示的文字，如在搜索结果中高亮显示搜索关键词
time	行内元素，定义日期或时间
meter	行内元素，定义已知最大值和最小值的度量项。必须定义独立的范围
progress	定义运行中的进程。可以用来显示 JavaScript 中耗费时间的函数的进程
audio	定义页面中插入的音频文件。
video	定义页面中插入的视频文件。
details	定义用户要求得到并且可以得到的细节信息，与 summary 标签配合使用。summary 标签是标题或图例且是 details 标签中的第一个子标签，在浏览器中可见。单击标题或图例时，就会显示出 details 里的内容。如：<details><summary>标题</summary>单击标题才可见的内容</details>
datagrid	定义可选数据的列表，通常用于显示树列表
menu	定义菜单列表，通常用于列出表单控件。其中的列表项用 标签定义
command	定义命令按钮，如单选框、复选框或普通按钮。例如：<command onclick＝"cut()" label＝"cut">

除 command 标签外，其他新增的标签都是双标签。

2. HTML 5 被取缔的标签

HTML 5 被取缔的标签及描述见表 2-12。

表 2-12　HTML 5 被取缔的标签及描述

类　　型	取缔的标签
能使用 CSS 代替的标签	basefont、big、center、font、s、strike、tt、u 这些标签的功能都可以通过 CSS 样式的设置来实现
不再使用 frame 框架	HTML 5 只支持 iframe 框架或服务器方创建的由多个页面组成的复合页面，因此，frameset、frame 和 noframes 这 3 个标签被取缔
只有部分浏览器支持的标签	applet、bgsound、blink、marquee 这些标签只有部分浏览器支持，在 HTML 5 中被取缔。applet 被 embed 标签代替，bgsound 被 audio 标签代替，marquee 可以由 JavaScript 程序实现

3. HTML 5 新增的属性

HTML 5 中新增的属性主要以表单属性为主，还有链接的相关属性(见表 2-13)及其他

属性(见表 2-14)。新增的表单属性将在后面的 HTML 5 表单中进行介绍。

表 2-13　新增的与链接相关的属性

属　性	描　　述
media	为<a>与<area>标签增加的属性。规定目标 URL 是以什么类型的媒介进行优化的。该属性只能在 href 属性存在时使用
hreflang/rel	为<area>标签增加的属性,保持与<a>标签和<link>标签的一致
sizes	为<link>标签增加的属性,可以与<icon>标签结合使用,指定关联图标(<icon>标签)的大小
target	为<base>标签增加的属性,主要是保持与<a>标签的一致。target 属性在与 iframe 结合使用时非常有用

表 2-14　HTML 5 新增的其他属性

属　性	描　　述
reversed	为 标签增加的属性,指定列表倒序显示
charset	为<meta>标签增加的属性,指定文档的字符编码
type label	为<menu>标签增加的两个属性。label 属性为菜单定义了一个可见的标注,type 属性使菜单有 3 种形式:上下文菜单、工具条和列表菜单
scoped	为<style>标签增加的属性,规定样式的作用范围。为<script>标签增加了 aync 属性,定义脚本是否异步执行
manifest	为<html>标签增加的属性,开发离线 Web 应用程序时与 API 结合使用,定义一个 URL,在这个 URL 上描述了文档的缓存信息
sandbox seamless srcdoc	为<iframe>标签增加的 3 个属性,用来提高页面的安全性,防止不信任的 Web 页面执行某些操作

对于 HTML 5 新增的功能,各个浏览器的支持程度不一致,主要是以新出的高版本浏览器为主。IE 9 及以上的版本支持 HTML 5 的浏览,本书则以火狐浏览器作为案例的展示。

4. HTML 5 中被取消的属性

HTML 5 中取消了一部分以往版本中常用的属性,主要以能被其他的标签或样式设置所代替的为主,如 align、bgcolor、text、valign 等属性。

2.4.3　HTML 5 音视频

在以往的网页中要播放音视频,浏览器都必须要安装相应的插件才能够播放,但是并不是所有的浏览器都有相应的插件。因此,与 HTML 4 相比,HTML 5 新增了音频标签和视频标签,规定了一种包含音视频的标准方法,不需要浏览器再安装其他相应的插件。

1. 音频标签

<audio></audio>标签是 HTML 5 中新增的标签,用来播放声音文件。支持三种音频格式:ogg、mp3 和 wav。

如果要在 HTML 5 页面中播放音频，需要使用<audio>标签的 src 属性指定要插入音频文件的源目标地址。<audio>标签的常用属性及描述见表 2-15。

表 2-15　<audio>标签的常用属性及描述

属　性	值	描　　述
src	URL	目标音频文件的地址
autoplay	autoplay(自动播放)	音频就绪后马上播放
controls	controls(控制)	提供添加播放、暂停和音量的控件
loop	loop(循环)	音频重复播放
preload	preload(加载)	音频在页面加载时进行加载，并预备播放
autobuffer	autobuffer(自动缓冲)	确定在显示网页时播放文件是由浏览器自动缓冲的，还是由用户使用相关 API 进行缓冲

<audio>标签还可以通过 source 属性添加多个音频文件。在<audio>与</audio>之间插入的内容是提供给不支持<audio>标签的浏览器显示的。

【例 2-9】　音频标签。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>添加音频文件</title>
</head>

<body>
    <audio controls="controls">
    </audio>
</body>
</html>
```

程序的运行结果如图 2-10 所示。



图 2-10　音频标签

2．视频标签

<video>标签是用来定义播放视频文件的，支持三种视频格式：ogg、mp4 和 WebM。与<audio>标签的用法一致，可以通过 source 属性添加多个视频文件。在<video>与

</video>之间插入的内容是为不支持<video>标签的浏览器显示的。<video>标签的常用属性及描述见表 2-16。

表 2-16　<video>标签的常用属性及描述

属　性	值	描　　述
src	URL	目标视频文件的地址
autoplay	autoplay(自动播放)	视频就绪后马上播放
controls	controls(控制)	提供添加播放、暂停和音量的控件
loop	loop(循环)	视频重复播放
preload	preload(加载)	视频在页面加载时进行加载,并预备播放。如果使用 autoplay,则忽略该属性
width	宽度值	设置视频播放器的宽度
height	高度值	设置视频播放器的高度
poster	url	当视频未响应或缓冲不足时,该属性值链接到一个图像。该图像将以一定比例显示

由于<video>标签与<audio>标签用法一致,这里就不再举例了。

2.4.4　HTML 5 表单

在 Web 应用中,经常会用到表单来设计登录或注册页面。在 HTML 5 中,表单的功能得到了加强,增加了一些属性及表单控件的类型,使表单的开发更快捷、更方便。

1. 新增的表单控件

在 HTML 5 中新增了一些表单的控件,主要是<input>标签中 type 属性值(即输入类型)的增加。如表 2-17 所示,列出了 HTML 5 中新增的表单控件。

表 2-17　<input>标签新增的输入类型

输 入 类 型	描　　述
color	调色板控件,呈现为单行文本框,提供了一个颜色选择器
date	日期控件
datetime	日期和时间控件
datetime-local	本地日期和时间控件
time	时间控件
month	月份控件
week	星期控件
range	滑动刻度控件
search	搜索文本框,一般在文本框中显示一个关闭符号
tel	单行文本框,用来输入电话号码的文本框
email	一个单行文本框,呈现电子邮件
url	单行文本框,用来输入一个完整的 URL 地址,包括传输协议
number	表现为一个单行文本框,或带步进按钮

这些输入类型全部都是<input>标签中的 type 属性值,且都是 HTML 5 中新增的输

入类型。对于这些新增的表单控件，不管浏览器支持与否都可以使用。如果浏览器不支持，则会直接忽略，不会报错。

(1) color 输入类型

该控件用于设置一个颜色的选择框，如：＜input type＝"color" name＝"user_color"＞。在 Firefox 浏览器中显示出一个具有默认值为"#000000"的黑色颜色框，当单击这个颜色选择框后，弹出一个颜色选择器，如图 2-11 所示。用户可以在颜色选择器中选择自己需要的颜色。在不支持该输入类型的浏览器中则会被忽略，形成一个普通的单行文本输入框。



图 2-11　color 输入类型在火狐浏览器中的显示效果

这里的颜色是用一个 RGB 的颜色值表示的颜色。

(2) 日期和时间控件

HTML 5 中提供了多种选择日期和时间的输入类型，用于验证输入的日期。

- date：用于选取年、月、日。如＜input type＝"date" name＝"user_date"＞。
- datetime：用于选取 UTC 时间，包括年、月、日、小时和分钟。如＜input type＝"datetime" name＝"user_datetime"＞。但在浏览器显示该类型的日期输入类型时，呈现出的是一个单行文本输入框，并不能选择日期，而且也不对日期格式进行验证。
- datetime-local：用于选取本地时间，包括年、月、日、小时和分钟。如＜input type＝"datetime-local" name＝"datetime_local"＞。
- time：用于选取时间、小时和分钟。如＜input type＝"time" name＝"user_time"＞。
- month：用于选取年和月。如＜input type＝"month" name＝"user_month"＞。
- week：用于选取年和第几周。如＜input type＝"week" name＝"user_week"＞。

日期选择器所涉及的这几种选取日期的输入类型，在 Chrome 浏览器中的显示效果如图 2-12 所示。

在 Firefox 浏览器中则会被忽略，形成普通的单行文本输入框，如图 2-13 所示。

其中可以选择月和日的日期类型的输入框，虽然会验证日期的格式，但是不会验证日期的准确性，也就是不会判断月份的大小月以及二月是否是闰月，所有的月份都是 31 天。每



图 2-12　日期和时间控件在 Chrome 浏览器中的显示效果

图 2-13　日期和时间控件在火狐浏览器中的显示效果

个能够选取日期类型的输入框右侧都具有一个删除图标按钮，一个附带步进按钮，一个单击后会弹出日期选择项的下拉框图标。选取的日期可以通过"删除图标"按钮删除，也可以通过步进按钮选择或者通过下拉框中的日期控件选择。

（3）range 输入类型

用于设定一定范围内的数字值，通常表现为一个滑动条。可以用 min 属性设置最小值，用 max 属性设置最大值。如<input type="range" name="price" min="10" max=

"20">。在 Firefox 浏览器中的显示效果如图 2-14 所示。



图 2-14　range 输入类型在火狐浏览器中的显示效果

当拖动滑块到一个位置后，提交表单，提交的数据中会显示出所拖动滑块拖动到的位置数值。

(4) search 输入类型

用于设定搜索框，如关键词搜索<input type="search" name="key_words">，search 类型显示出一个单行文本框的形式。在 Firefox 浏览器中显示的搜索框与普通的文本输入框无异，如要查看显示效果，请查看并运行例 2-10 以观效果。

(5) tel 输入类型

用于定义一个电话号码的输入框，当鼠标光标移到输入框上面会显示一个信息提示框。但是电话号码的形式多种多样，很难有一个固定的模式。因此，仅仅用 tel 类型来定义电话号码是无法实现的，通常与 pattern 属性结合使用，利用 pattern 属性的正则表达式来规定电话号码的格式。如<input type="tel" name="phone_number" pattern="^\d{11}$">定义了一个必须具有 11 位数字的手机号码输入框，当格式不正确时，输入框的边框会显示为红色，如图 2-15 所示。

(6) email 输入类型

用于设置一个输入邮箱地址的输入框，当鼠标光标移到输入框上面会显示一个信息提示框。如<input type="email" name="user_email">。当在设置为 email 类型的输入框中输入一个不是邮箱地址的字符时，输入框的边框会显示为红色，如图 2-16 所示。



图 2-15　tel 输入类型在火狐浏览器中的显示效果

图 2-16　email 输入类型在火狐浏览器中的显示效果

email 输入类型输入框的验证只能简单验证用户在输入框中输入的电子邮件的地址是否包含了"@"符号且该符号不在第一个字符，不会真正验证电子邮件地址格式的正确性，即电子邮件地址写成"123@"会被判断成正确的。

(7) URL 输入类型

用于设置一个 URL 地址的输入框，当鼠标移到输入框上面会显示一个提示信息框。在该输入框中输入的内容必须是一个绝对的 URL 地址，否则输入框的边框就会显示为红色。如<input type="url" name="user_url">。在 Firefox 浏览器中的显示效果如图 2-17 所示。



图 2-17　URL 输入类型在火狐浏览器中的显示效果

这个地址输入框在输入 URL 地址时必须输入一个完整的绝对 URL 地址，包括传输协议，否则就会报错。只要有传输协议则不会出任何问题。传输协议可以使用 HTTP 或者 FTP，本地的绝对 URL 地址也可以，即地址开头为"file:"。

(8) number 输入类型

用于一个设置数值的输入框，当鼠标光标移到输入框上面会显示一个信息提示框。可

以对输入的数值设定一个范围，分别用 min 属性设置最小数值，用 max 属性设置最大数值。如<input type="number" name="use_age" min="14" max="100">，在 Firefox 浏览器中的显示效果如图 2-18 所示。

图 2-18　number 输入类型在火狐浏览器中的显示效果

在 Firefox 浏览器中，当输入框失去焦点时会自动验证输入的值是否在控件所设置的限定范围内。如果不在设置的范围内，输入框失去焦点时，它的边框就会显示为红色，表示输入错误。

以上所列举的 tel、email、url 和 number 输入类型在 Firefox 浏览器中的验证错误都是以输入框失去焦点时，它的边框显示成红色为格式错误提示信息。在 Chrome 浏览器中则是以提交表单时验证输入值是否符合规范，如果不符合，提交时会在输入框位置弹出一个提示框显示规范要求信息。如果读者需要查看不同浏览器的运行效果，请自行运行例 2-10 的代码以查看运行效果。

【例 2-10】 HTML 5 新增表单控件。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>新增表单控件</title>
</head>
<body>
    <form>
      color 输入类型：<input type="color" name="user_color"><br>
      date 输入类型：<input type="date" name="user_date"><br>
      datetime 输入类型：<input type="datetime" name="user_datetime"><br>
      datetime-local 输入类型：<input type="datetime-local" name="datetime_
      local"><br>
      time 输入类型：<input type="time" name="user_time"><br>
      month 输入类型：<input type="month" name="user_month"><br>
      week 输入类型：<input type="week" name="user_week"><br><br>
      range 输入类型：<input type="range" name="price" min="10" max="20"><br>
      <br>
      search 输入类型：<input type="search" name="key_words"><br><br>
      电话号码：<input type="tel" name="phone_number" pattern="^\d{11}$"><br>
      <br>
      电子邮箱：<input type="email" name="user_email"><br><br>
      输入 URL 地址：<input type="url" name="user_url"><br><br>
      年龄：<input type="number" name="use_age" min="14" max="100">
      <input type="submit">
    </form>
</body>
</html>
```

2. 新增的表单属性

HTML 5 新增的表单属性见表 2-18。

表 2-18　HTML 5 新增的表单属性

属　　性	描　　述
placeholder	在输入型文本框中显示描述性说明或提示信息。输入框中一旦有用户的输入值后，描述性说明或提示信息就会消失
autocomplete	是否保存输入值以备将来使用。值 on 表示保存，值 off 表示不保存
autofocus	当页面打开时，确定表单控件是否获取光标焦点。一般一个页面只有一个表单控件可以设置该属性
list	为单行文本框增加的属性，属性值为某个 datalist 标签的 id 名，形成一个类似于下拉选择框的控件，可以直接从选择框中选择输入值。当选择列表中没有输入值时，用户可以自行输入
min/max	为 range 控件增加的属性，限定数值的输入范围，min 为设置的最小值，max 为设置的最大值
step	为 range 控件增加的属性，设置输入值递增或递减的梯度
required	设置输入型控件为必填项，否则无法提交表单。该属性是表单中一种最简单的验证方式

针对以上的 list 属性，通过例 2-11 进行详细说明。

【例 2-11】　list 属性的应用。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>list 属性</title>
</head>

<body>
    学校<input type="text" list="school">
    <datalist id="school" style="display:none;">
        <option value="重庆工程学院">重庆工程学院</option>
        <option value="重庆大学">重庆大学</option>
        <option value="重庆工商大学">重庆工商大学</option>
    </datalist>
</body>
</html>
```

程序的运行结果如图 2-19 所示。

3. HTML 5 新增的<output>标签

<output>标签是 HTML 5 中新增的一个表单控件，用来显示计算结果或脚本的运行结果。该标签必须包含在某个表单中，或者给它添加 form 属性。



图 2-19　list 属性

【例 2-12】 <output>标签。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>output 标签</title>
    <script>
        function res(){
            document.getElementById("jieguo").value=document.getElementById
            ("shu").value;
        }
    </script>
</head>

<body>
    <form oninput="res()">
        请拖动滑块选择一个数值：
        <input type="range" min="0" max="100" step="5" id="shu">
        <output id="jieguo">50</output>
    </form>
</body>
</html>
```

程序的运行结果如图 2-20 所示。



图 2-20　应用<output>标签后的显示效果

当拖动滑块时，<output>标签所在的表单就会接收到用户的输入信息，并将脚本运行结果显示在<output>标签中。

4. 表单的自动验证

在 HTML 5 中，表单可以实现简单的自动验证。对于表单控件实现的简单自动验证功能基本上是由一些新增的表单控件属性实现的。下面介绍几个常用的自动验证属性。

(1) required 属性

required 属性设置输入型表单控件为必填项。表单提交时，如果设置了 required 属性的表单控件内容为空，则不允许提交，且会出现提示用户出错的信息。

(2) pattern 属性

pattern 属性是为表单控件设置正则表达式所使用的属性。提交表单时，会自动验证表单控件中的输入值是否符合正则表达式所规定的格式规则，如果不符合格式规则，则不允许提交表单，并会有错误提示。

(3) max 属性和 min 属性

max 和 min 这两个属性主要用在 range 输入类型和日期类型的表单控件中，限定其输入数字或日期的范围，超出该范围，则表单提交时会自动验证并报错。

(4) step 属性

step 属性限定输入数值递增或递减的梯度，如果不符合该梯度值的设置，表单提交时同样会自动验证并报错。

2.4.5　HTML 5 画布

在 HTML 5 中新增了很多的功能，其中最显著的功能就是可以直接在 HTML 页面中绘制图形。这里所说的绘制图形并不是用鼠标绘画，而是通过 HTML 5 新增的<canvas>标签创建一块矩形区域，然后再通过 JavaScript 控制添加图片或绘制线条、文字和图形，甚至可以加入动画。<canvas>标签所创建出来的这块区域被称为画布。

1. <canvas>标签

<canvas>标签是 HTML 5 中新增的一个很重要的标签，专门用来在页面上绘制图形。但实际上<canvas>标签自身并不绘制图形。它相当于一张空白的画布，如果需要绘制图形，必须使用 JavaScript 脚本进行绘制。创建一对<canvas></canvas>标签，即在页面中放置了一个可以绘图的画布，该画布可以通过样式来设置区域的宽度和高度。

【例 2-13】 <canvas>标签创建绘图画布。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>canvas 标签</title>
</head>

<body>
```

```
    <canvas id="mycanvas" width="200" height="200" style="border:1px solid #
   000;">
    </canvas>
</body>
</html>
```

程序的运行结果如图 2-21 所示。

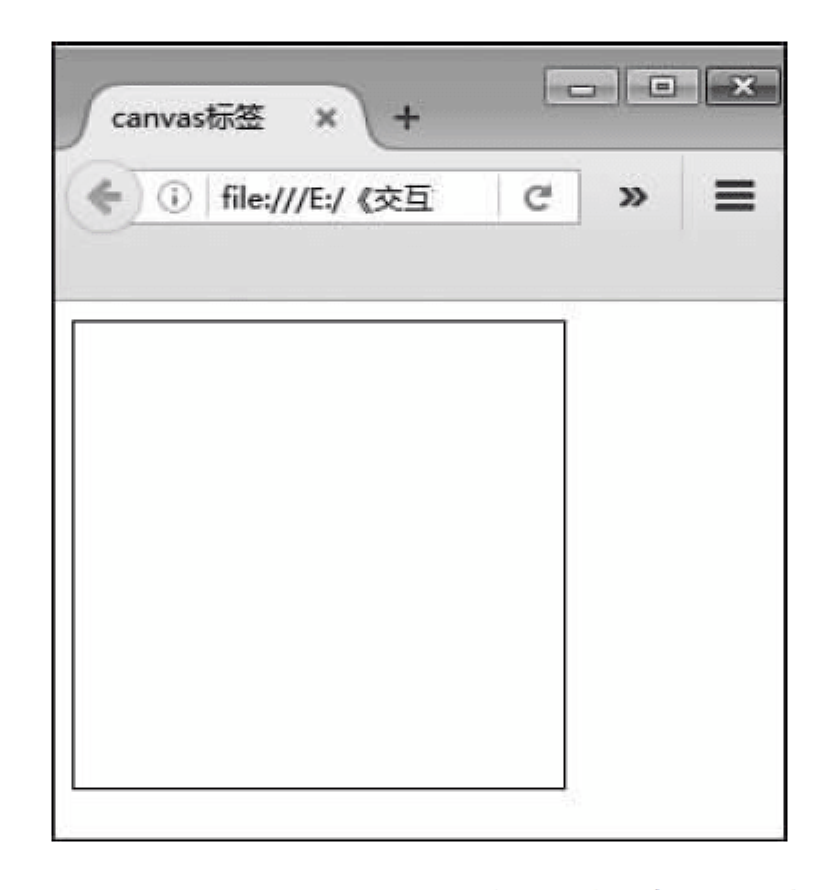


图 2-21 用<canvas>标签创建的画布在火狐浏览器中的显示效果

在图 2-21 中创建了一个宽度为 200 像素、高度为 200 像素的画布。由于创建的画布只是一个矩形区域，没有任何特征，在浏览器中只会显示为空白，看不到效果。因此，通过样式设置给它添加了一个黑色边框，使其能够展现在我们眼前，从而看见画布的区域范围。另外为该<canvas>标签设置了一个 ID 名，主要是因为<canvas>标签内绘制图形都是通过 JavaScript 脚本进行控制的。没有 ID，JavaScript 程序代码要找到<canvas>标签这个对象是很困难的，因此我们在创建<canvas>对象时都会给它添加一个 ID 名称以便通过脚本查找对象。

2. 绘制图形

在<canvas>标签中绘制图形的步骤如下。

第一步：获取<canvas>标签对应的 DOM 对象，这是一个 canvas 对象。

第二步：调用 canvas 对象的 getContext()方法，返回一个 CanvasRenderingContext2D 对象，用该对象即可绘制图形。

第三步：调用 CanvasRenderingContext2D 对象的方法绘制图形。

在以上的步骤中，canvas 对象的 getContext()方法用来选择在画布上绘制图形的类型。这里有“2d”和“3d”可供选择，但只有“2d”是当前的唯一合法值，可指定为二维绘图。将来如果<canvas>标签扩展到支持 3D 绘图，getContext()方法可能允许传递一个“3d”字符串参数来进行 3D 绘图。getContext()方法的返回值是一个 CanvasRenderingContext2D 对象，使用它可以在<canvas>标签对象中绘图。

用<canvas>标签绘制图形有两种方式：填充(fill)和绘制边框(stroke)。填充是填满图形的内部区域；绘制边框是不填充图形的内部，只绘制图形的边框。

这两种绘制图形的方式分别有各自的样式设置：填充图形方式的样式采用 fillStyle 属性设置；绘制图形边框方式的样式采用 strokeStyle 属性设置；线条的粗细样式采用 lineWidth 属性设置。

(1) 绘制矩形

在 canvas 对象中绘制矩形有两种方式，即填充一个矩形区域和绘制一个矩形边框。表 2-19 中分别列举了矩形绘制两种方式的定义。

表 2-19　矩形绘制两种方式的定义

方　　式	定义与描述
填充矩形区域	context.fillRect(x,y,width,height)填充一个起点坐标为(x,y)、宽度为 width、高度为 height 的矩形区域
绘制矩形边框	context.strokeRect(x,y,width,height)绘制一个起点坐标为(x,y)、宽度为 width、高度为 height 的矩形边框

以上定义中,context 指的是 getContext()方法返回的 CanvasRenderingContext2D 对象,画布的左上角为坐标原点,矩形的起点为矩形的左上角。

【例 2-14】　绘制矩形。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>绘制矩形</title>
</head>
<body>
    <canvas id="myrect" width="400" height="200" style="border:1px solid
    #000;"></canvas>
    <script type="text/javascript">
        var cvs=document.getElementById("myrect");
        var context=cvs.getContext("2d");
        context.fillRect(50,50,120,80);
        context.strokeRect(200,50,120,80);
    </script>
</body>
</html>
```

程序的运行结果如图 2-22 所示。

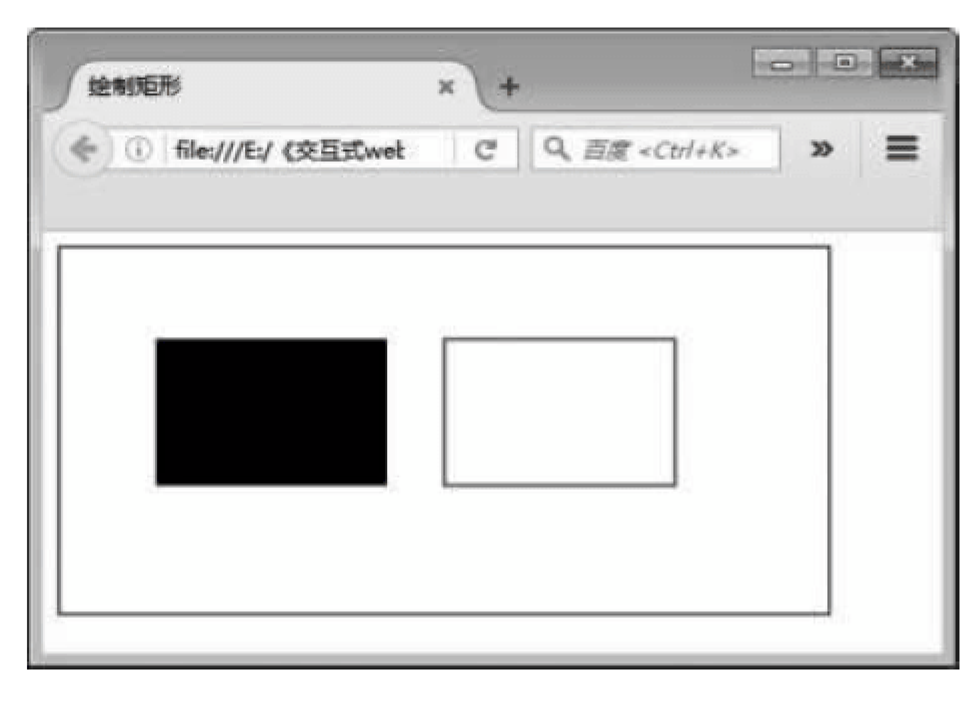


图 2-22　绘制矩形

例 2-14 中通过 fillRect(50,50,120,80)和 strokeRect(200,50,120,80)填充了一个起点坐标为(50,50)、宽为 120 像素、高为 80 像素的矩形区域,和绘制了一个起点坐标为(200,50)、宽为 120 像素、高为 80 像素的矩形边框。在没有样式设置的情况下,默认的填充颜色和线条颜色均为黑色。如果要对绘制的图形设置样式,可以通过表 2-20 中的方法设置。

表 2-20　绘制图形样式来设置图形的属性

方　法	定义与描述
fillStyle()	填充图形样式的设置，主要设置颜色
strokeStyle()	绘制图形边框样式的设置，主要设置颜色

绘制图形时，填充颜色和边框的颜色分别通过 fillStyle 属性和 strokeStyle 属性来设置。颜色值可以使用普通的英文颜色名称、样式表中的十六进制颜色值或者 rgb(0～255，0～255，0～255)函数、rgba(0～255，0～255，0～255，0～1)函数来表示。

【例 2-15】 绘制具有颜色样式的矩形。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>绘制具有颜色样式的矩形</title>
</head>
<body>
    <canvas id="myrect" width="400" height="200" style="border:1px solid
    #000;"></canvas>
    <script type="text/javascript">
        var cvs=document.getElementById("myrect");
        var context=cvs.getContext("2d");
        context.fillStyle="#f0f";
        context.fillRect(50,50,120,80);
        context.fillStyle="rgba(255,0,0,0.6)";
        context.fillRect(30,70,120,80);
        context.strokeStyle="rgb(0,255,0)";
        context.lineWidth=10;
        context.strokeRect(200,50,120,80);
        context.strokeStyle="yellow";
        context.lineJoin="round";
        context.strokeRect(185,70,120,80);
    </script>
</body>
</html>
```

程序的运行结果如图 2-23 所示。

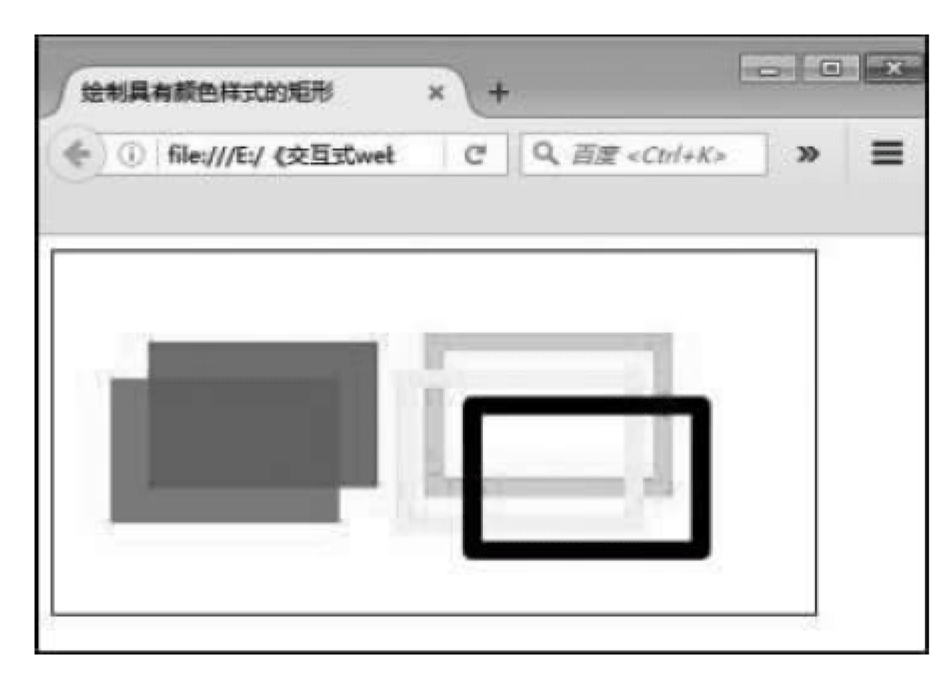


图 2-23　绘制具有颜色样式的矩形

例 2-15 中通过 fillStyle 属性设置了矩形的填充颜色，strokeStyle 属性设置了矩形的边框颜色。这两个属性都可以多次设置，绘制图形的颜色取决于在绘制图形前的最后一次颜色设置。颜色可以设置透明度。案例中的 lineWidth 属性是用来设置边框线条的宽度。lineJoin 属性设置线条连接点的风格，lineJoin 属性值可以为 meter（默认）、round（圆）、bevel（斜面）。

（2）绘制字符

在 canvas 对象中绘制字符类似于绘制矩形，同样有两种方式，即 fillText()填充字符和 strokeText()绘制字符轮廓。定义格式如下：

```
fillText(text,x,y[,maxWidth]);
strokeText(text,x,y[,maxWidth]);
```

其中参数 text 为绘制的文字内容；(x，y)为绘制字符的起点坐标，正常情况下，以第一个字符的左上角为起点坐标；maxWidth 为绘制字符的最大宽度。

样式设置还是采用 fillStyle 属性和 strokeStyle 属性；字符的字体样式通过 font 属性设置，并且按照 font＝"fontStyle fontVariant fontWeight fontSize/lineHeight fontFamily"的顺序进行设置；字符的水平对齐方式通过 textAlign 属性设置，如图 2-24 所示；字符的垂直对齐方式通过 textBaseline 属性设置，如图 2-25 所示。

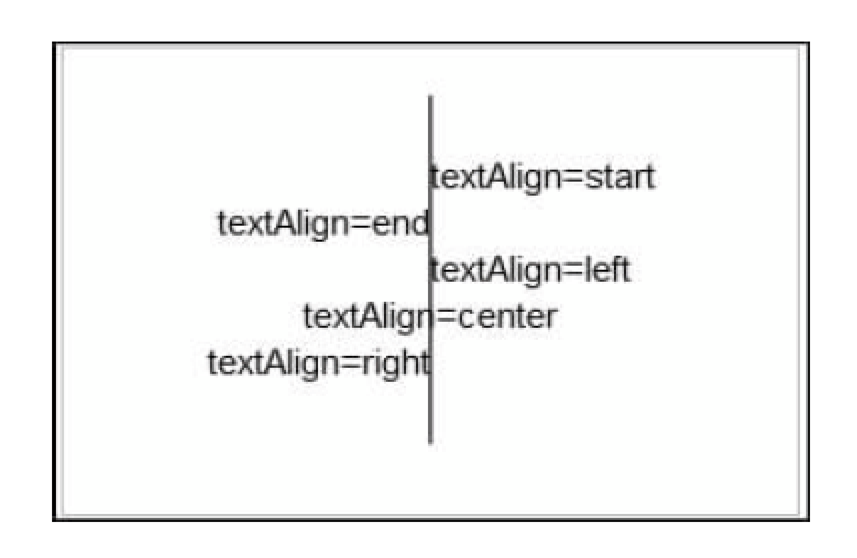


图 2-24　绘制字符的水平对齐方式

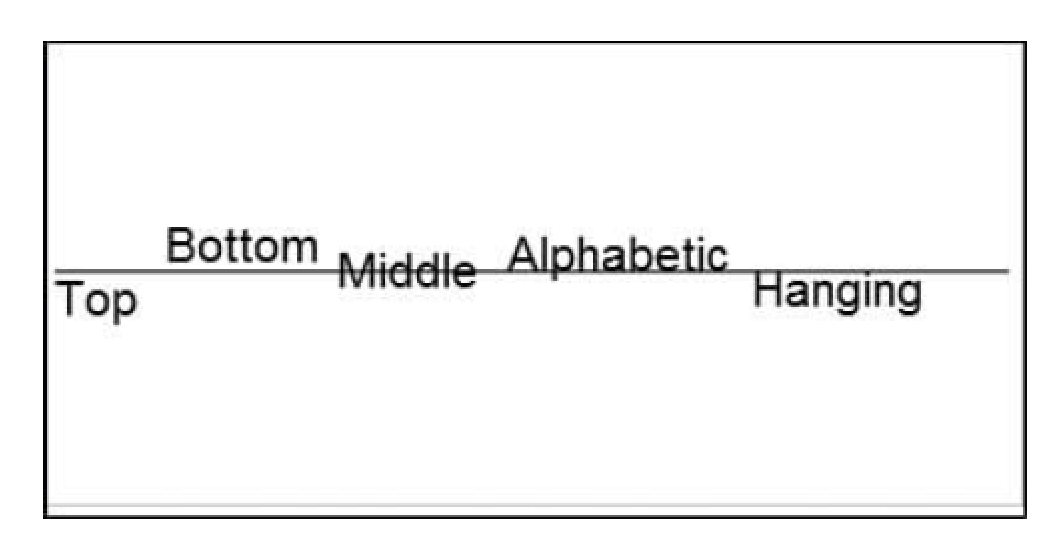


图 2-25　绘制字符的垂直对齐方式

【例 2-16】　绘制字符。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>绘制字符</title>
</head>

<body>
    <canvas id="mystr" width="500" height="300" style="border:1px solid
    #000;"></canvas>
    <script type="text/javascript">
        var cvs=document.getElementById("mystr");
        var str=cvs.getContext("2d");
        str.fillStyle="rgb(255,0,0)";
        str.font="normal bold 50px 隶书";
```

```
        str.textAlign="left";
        str.textBaseline="top";
        str.fillText("交互式 Web 前端开发",20,50,200);
        str.fillText("交互式 Web 前端开发",20,120);
        str.strokeText("交互式 Web 前端开发",20,200);
    </script>
</body>
</html>
```

程序的运行结果如图 2-26 所示。



图 2-26　绘制字符

(3) 设置阴影

在 canvas 对象中绘制图形时可以设置阴影。可设置阴影的属性及其描述如表 2-21 所示。

表 2-21　可设置阴影的属性及其描述

属　　性	描　　述
shadowBlur	设置阴影的模糊度,是一个浮点数,数值越大,模糊程度越大
shadowColor	设置阴影的颜色
shadowOffsetX	设置阴影在 X 方向的偏移
shadowOffsetY	设置阴影在 Y 方向的偏移

【例 2-17】　设置阴影。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>设置阴影</title>
```

```
</head>

<body>
    <canvas id="mycvs" width="500" height="300" style="border:1px solid
    #000;"></canvas>
    <script type="text/javascript">
        var cvs=document.getElementById("mycvs");
        var context=cvs.getContext("2d");
        context.fillStyle="rgba(255,0,0,1)";
        context.shadowBlur=15;
        context.shadowColor="#000";
        context.shadowOffsetX=-10;
        context.shadowOffsetY=-6;
        context.fillRect(80,30,120,80);
        context.strokeRect(280,30,120,80);
        context.font="normal bold 50px 隶书";
        context.shadowColor="#f0f";
        context.fillText("交互式 Web 前端开发",20,190);
        context.strokeText("交互式 Web 前端开发",20,270);
    </script>
</body>
</html>
```

程序的运行结果如图 2-27 所示。



图 2-27　设置阴影

3. 使用路径绘制图形

在<canvas>标签中使用路径可以绘制更复杂的图形。使用路径绘制图形的步骤如下。

第一步：调用 CanvasRenderingContext2D 对象的 beginPath()方法开始定义路径。

第二步：调用 CanvasRenderingContext2D 的各种方法添加子路径。

第三步：调用 CanvasRenderingContext2D 的 closePath()方法关闭路径。

第四步：调用 CanvasRenderingContext2D 的 fill()或 stroke()方法来填充或绘制路径。

(1) 绘制直线

使用路径绘制直线将会使用到 canvas 对象中的一些方法，如表 2-22 所示。

表 2-22　使用路径的方法及描述

方　法	描　　述
beginPath()	开始一条路径，或重置当前的路径
closePath()	创建从当前点到指定点的路径
moveTo()	移动到一个指定的坐标。可以指定当前点的坐标
lineTo()	创建从当前点到画布中指定点的线条(该方法并不会创建线条)

上面的方法只是定义使用路径的方法，真正绘制确定的路径和图形却需要使用 stroke()方法在画布上绘制路径和使用 fill()方法来填充图像(默认是黑色)。

【例 2-18】 绘制矩形的对角线和使用路径绘制三角形。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>绘制直线</title>
</head>
<body>
    <canvas id="myline" width="400" height="300" style="border:1px solid
    #000;">
    </canvas>
    <script type="text/javascript">
        var cvs=document.getElementById("myline");
        var context=cvs.getContext("2d");
        context.strokeRect(50,20,120,80);
        context.beginPath();
        context.moveTo(50,20);
        context.lineTo(170,100);
        context.closePath();
        context.stroke();
        context.beginPath();
        context.moveTo(200,120);
        context.lineTo(300,250);
        context.lineTo(100,250);
        //context.lineTo(200,120);
        //context.closePath();
        context.lineWidth=5;
        context.stroke();
        context.fillStyle="rgba(255,0,0,0.6)";
        context.fill();
```

```
    </script>
</body>
</html>
```

程序的运行结果如图 2-28 所示。

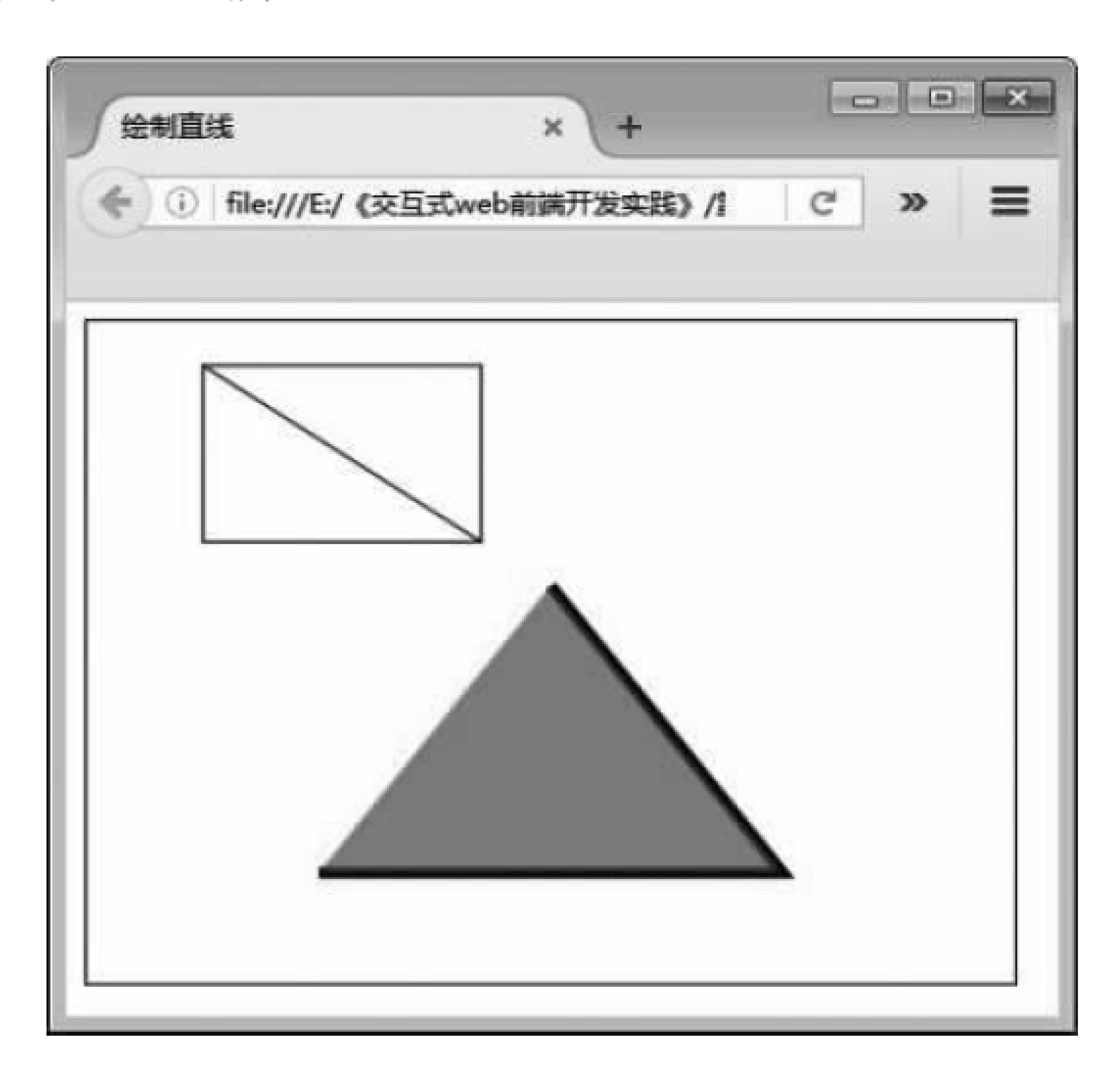


图 2-28　例 2-18 的运行结果

(2) 使用 arcTo()方法绘制圆弧

使用 arcTo()方法可以向 canvas 对象的当前路径上添加一段圆弧。格式如下：

```
arcTo(x1,y1,x2,y2,radius)
```

通过 arcTo()方法绘制圆弧的方式如下：假设从当前点到 $P_1(x_1, y_1)$绘制一条线条，再从 $P_1(x_1, y_1)$到 $P_2(x_2, y_2)$绘制一条线条，arcTo()则绘制一段同时与上面两条线相切且半径为 radius 的圆弧，如图 2-29 所示。

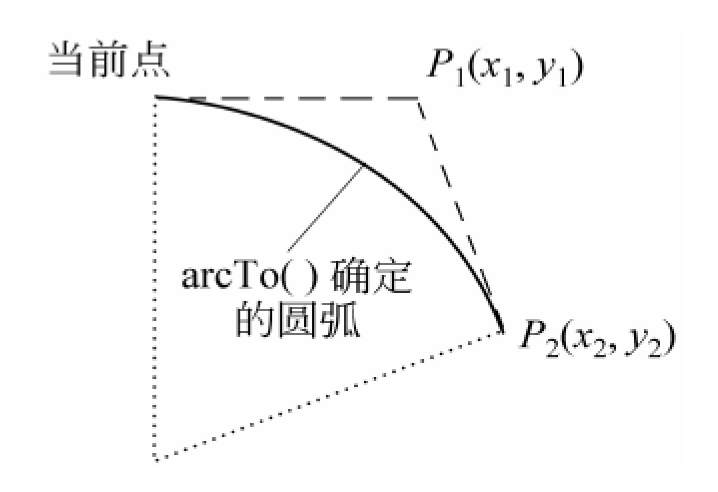


图 2-29　用 arcTo()方法绘制的圆弧

【例 2-19】　通过 arcTo()方法绘制一个圆角矩形。

```
<!doctype html>
<html>
<head>
```

```
    <meta charset="utf-8">
    <title>绘制圆角矩形</title>
</head>

<body>
    <canvas width="400" height="300" id="yjjx" style="border:1px solid #000;">
    </canvas>
    <script type="text/javascript">
        var djx=document.getElementById("yjjx").getContext("2d");
            //绘制圆角矩形
            /*
                该方法绘制一个圆角矩形
                x,y: 圆角矩形左上角的坐标
                width、height: 控制圆角矩形的宽、高
                radius: 控制圆角矩形四个圆角的半径
            */
            function creatroundrect(context,x,y,width,height,radius){
                context.beginPath();
                //移动到左上角
                context.moveTo(x+radius,y);
                //添加一条连接到右上角的线段
                context.lineTo(x+width-radius,y);
                //添加一段圆弧
                context.arcTo(x+width,y,x+width,y+radius,radius);
                //添加一条连接到右下角的线段
                context.lineTo(x+width,y+height-radius);
                //添加一段圆弧
                context.arcTo(x+width,y+height,x+width-radius,y+height,
                radius);
                //添加一条连接到左下角的线段
                context.lineTo(x+radius,y+height);
                //添加一段圆弧
                context.arcTo(x,y+height,x,y+height-radius,radius);
                //添加一条连接到左上角的线段
                context.lineTo(x,y+radius);
                //添加一段圆弧
                context.arcTo(x,y,x+radius,y,radius);
                context.closePath();
            }
            creatroundrect(djx,50,50,200,150,20);
            djx.lineWidth=5;
            djx.strokeStyle="#f0f";
            djx.stroke();
    </script>
</body>
</html>
```

程序的运行结果如图 2-30 所示。

(3) 使用 arc()方法绘制圆弧

使用 arc()方法也可以创建一段圆弧或曲线。格式如下：

```
arc(x,y,radius,startAngle,endAngle, [counterclockwise])
```

向 canvas 对象的当前路径上添加一段弧，绘制以(x,y)为圆心、radius 为半径，从 startAngle 角度开始，到 endAngle 角度结束的圆弧。startAngle、endAngle 以弧度作为单位。counterclockwise 参数为可选项，规定应该逆时针还是顺时针进行绘图。设置为 false 则顺时针绘制圆弧，设置为 true 则为逆时针绘制圆弧。如需使用 arc()方法来创建一个圆形，请把起始角弧度设置为 0，结束角弧度设置为 2 * Math.PI。使用 arc()方法绘制圆弧的示意图如图 2-31 所示。

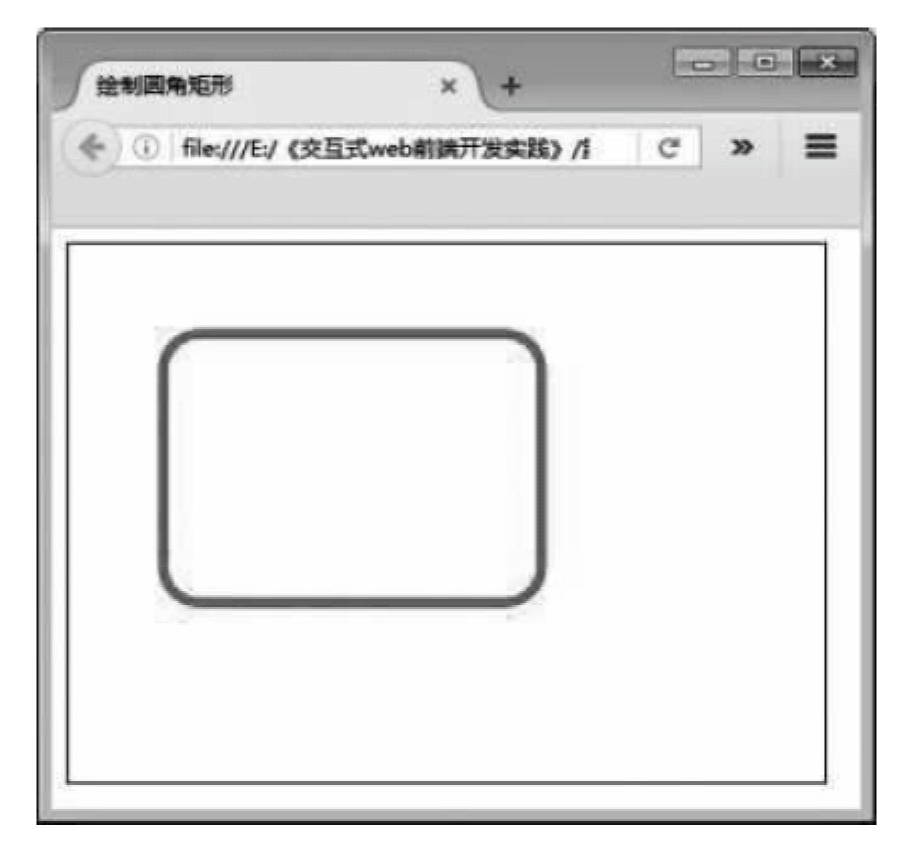


图 2-30 绘制圆角矩形

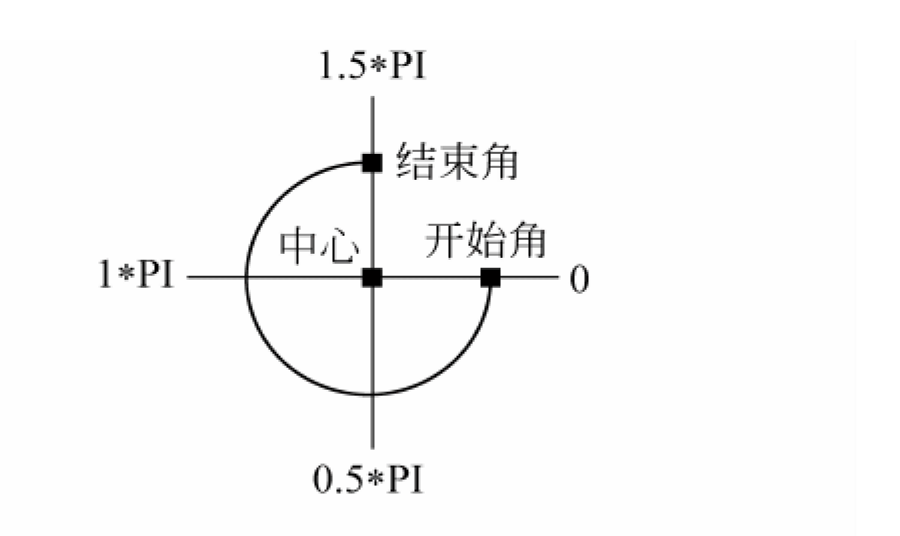


图 2-31 用 arc()方法绘制圆弧的示意图

【例 2-20】 通过 arc()方法绘制一个圆，1/4 圆弧和一个闭合的 3/4 圆形。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>arc()方法绘制圆及圆弧</title>
</head>
<body>
    <canvas width="400" height="300" id="mycircle" style="border:1px solid
    #000;"></canvas>
    <script type="text/javascript">
        var cvs=document.getElementById("mycircle");
        var context=cvs.getContext("2d");
        context.arc(100,100,75,0,2*Math.PI,true);
        context.fillStyle="rgba(255,0,0,0.6)";
        context.fill();
        context.beginPath();
        context.arc(300,100,75,0,1.5*Math.PI,true);
```

```
        context.stroke();
        context.beginPath();
        context.arc(250,200,75,0,1.5 * Math.PI,false);
        context.closePath();
        context.stroke();
    </script>
</body>
</html>
```

程序的运行结果如图 2-32 所示。

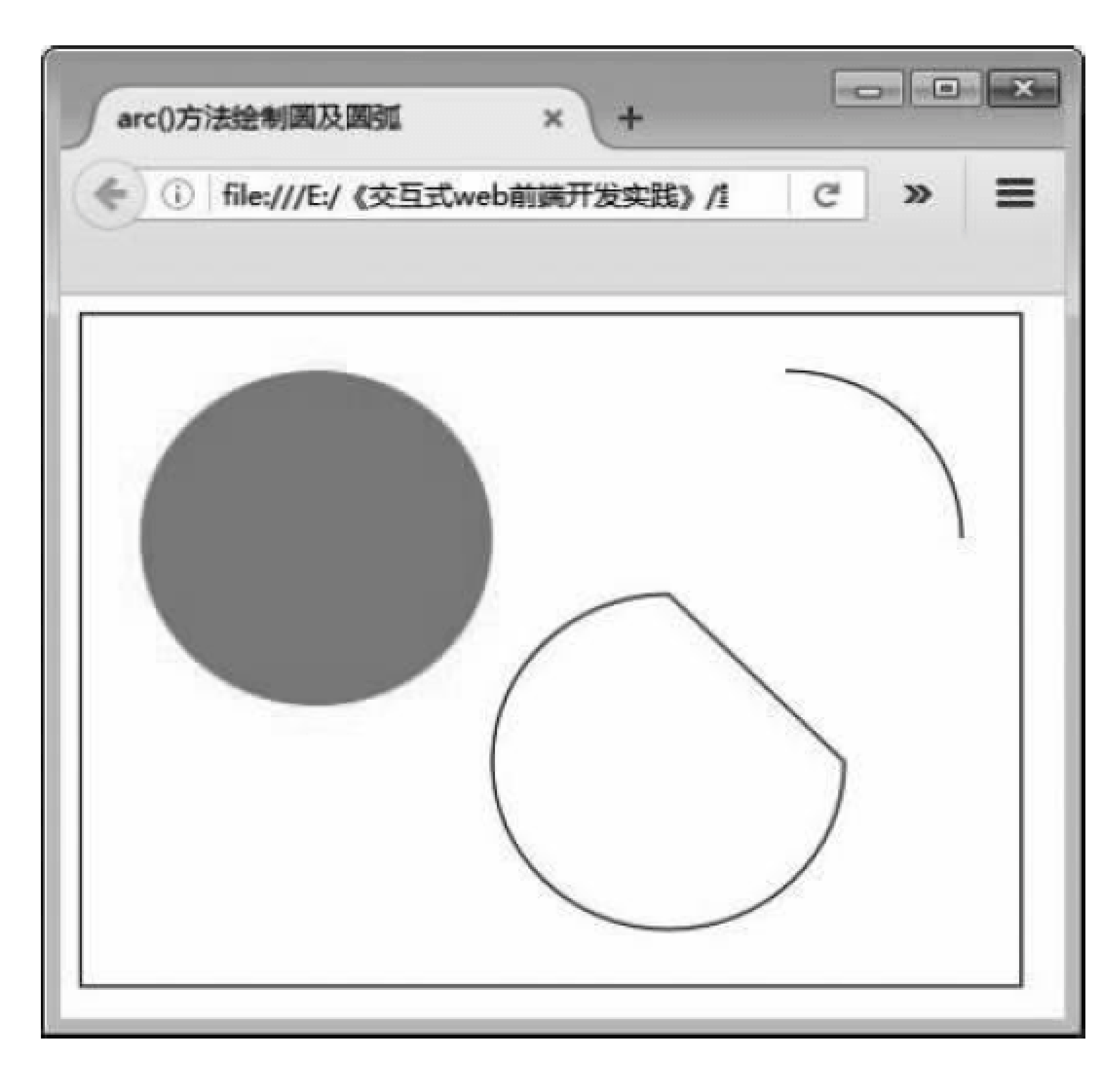


图 2-32　用 arc()方法绘制圆及圆弧

【例 2-21】 圆的投影。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>圆的投影</title>
</head>
<body>
    <canvas id="ty" width="400" height="300" style="border:1px solid #000;
    background:#000;">
    </canvas>
    <script type="text/javascript">
        var tycanvas=document.getElementById("ty");
        var ty=tycanvas.getContext("2d");
        for(var i=0;i<10;i++){
            ty.beginPath();
            ty.arc(i * 25,i * 25,(i+1) * 8,0,2 * Math.PI);
            ty.closePath();
```

```
            ty.fillStyle="rgba(255,0,255,"+(10-i)*0.1+")";
            ty.fill();
        }
    </script>
</body>
</html>
```

程序的运行结果如图 2-33 所示。

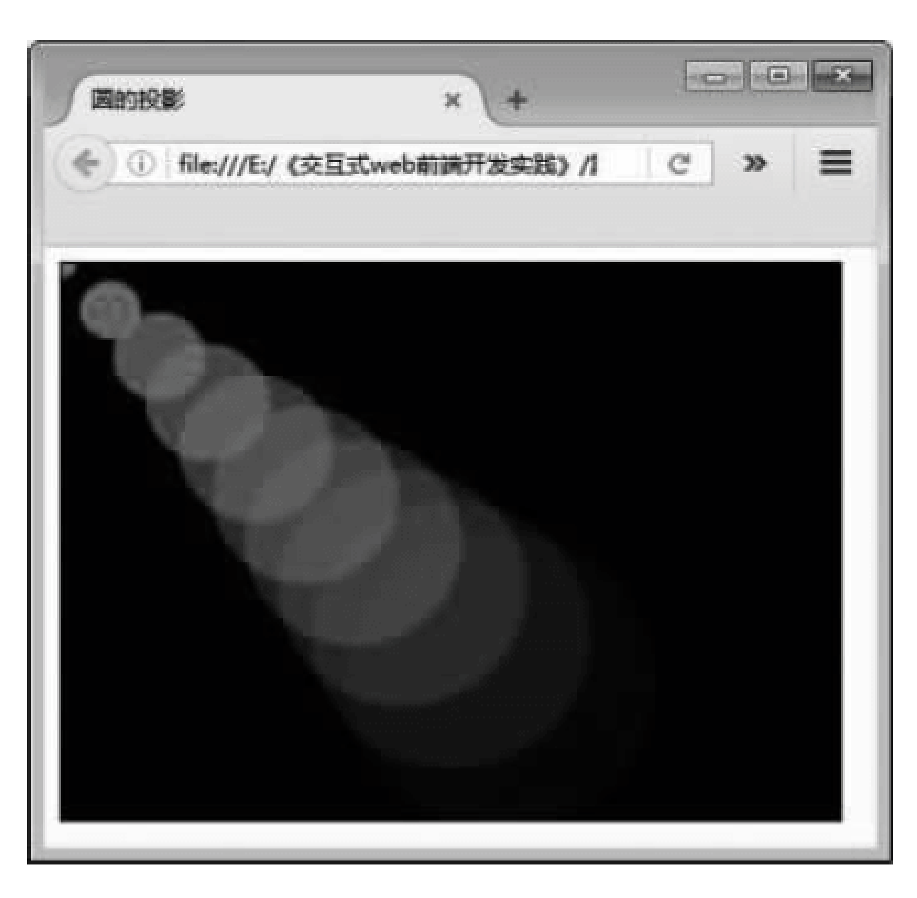


图 2-33　圆的投影

使用路径还可以画出多边形、花朵、桃心、雪花、五角星、火柴人等更加复杂的图形，这里就不一一举例了，留给读者自己独立完成。

4. 绘制颜色渐变的图形

在 canvas 对象上还可以绘制颜色渐变的图形。常用的有两种渐变形式：线性渐变和径向渐变。

（1）线性渐变

创建线性渐变的步骤如下。

第一步：调用 createLinearGradient(xstart，ystart，xend，yend)方法创建一个线性渐变，该方法返回一个 CanvasGradient 对象。

第二步：调用 CanvasGradient 对象的 addColorStop(float offset，string color)方法向渐变对象中添加颜色。参数 offset 控制添加颜色的点，是 0～1 之间的一个小数。0 表示把颜色添加在起始点，1 表示把颜色添加在结束点。参数 color 为添加的颜色值。

第三步：再将 CanvasGradient 对象赋值给 fillStyle 或 strokeStyle 属性即可。

【例 2-22】　线性渐变。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>线性渐变</title>
</head>

<body>
    <canvas id="linegrad" width="400" height="280" style="border:1px solid
    #000;"></canvas>
    <script type="text/javascript">
        var ctx=document.getElementById("linegrad").getContext("2d");
        ctx.save();
        ctx.translate(30,20);
        //创建线性渐变
        var lg=ctx.createLinearGradient(0,0,160,80);
        //向线性渐变上添加颜色
        lg.addColorStop(0.2,"#f00");
```

```
        lg.addColorStop(0.5,"#0f0");
        lg.addColorStop(0.9,"#00f");
        //赋值
        ctx.fillStyle=lg;
        ctx.fillRect(0,0,160,80);
        ctx.restore();
        ctx.translate(280,160);
        ctx.beginPath();
        ctx.arc(0,0,80,0,2*Math.PI,true);
        ctx.closePath();
        ctx.lineWidth=12;
        //创建线性渐变
        var lg2=ctx.createLinearGradient(-40,-40,80,50);
        //向线性渐变上添加颜色
        lg2.addColorStop(0.1,"#ff0");
        lg2.addColorStop(0.4,"#0ff");
        lg2.addColorStop(0.8,"#f0f");
        ctx.strokeStyle=lg2;
        ctx.stroke();
    </script>
</body>
</html>
```

程序的运行结果如图 2-34 所示。

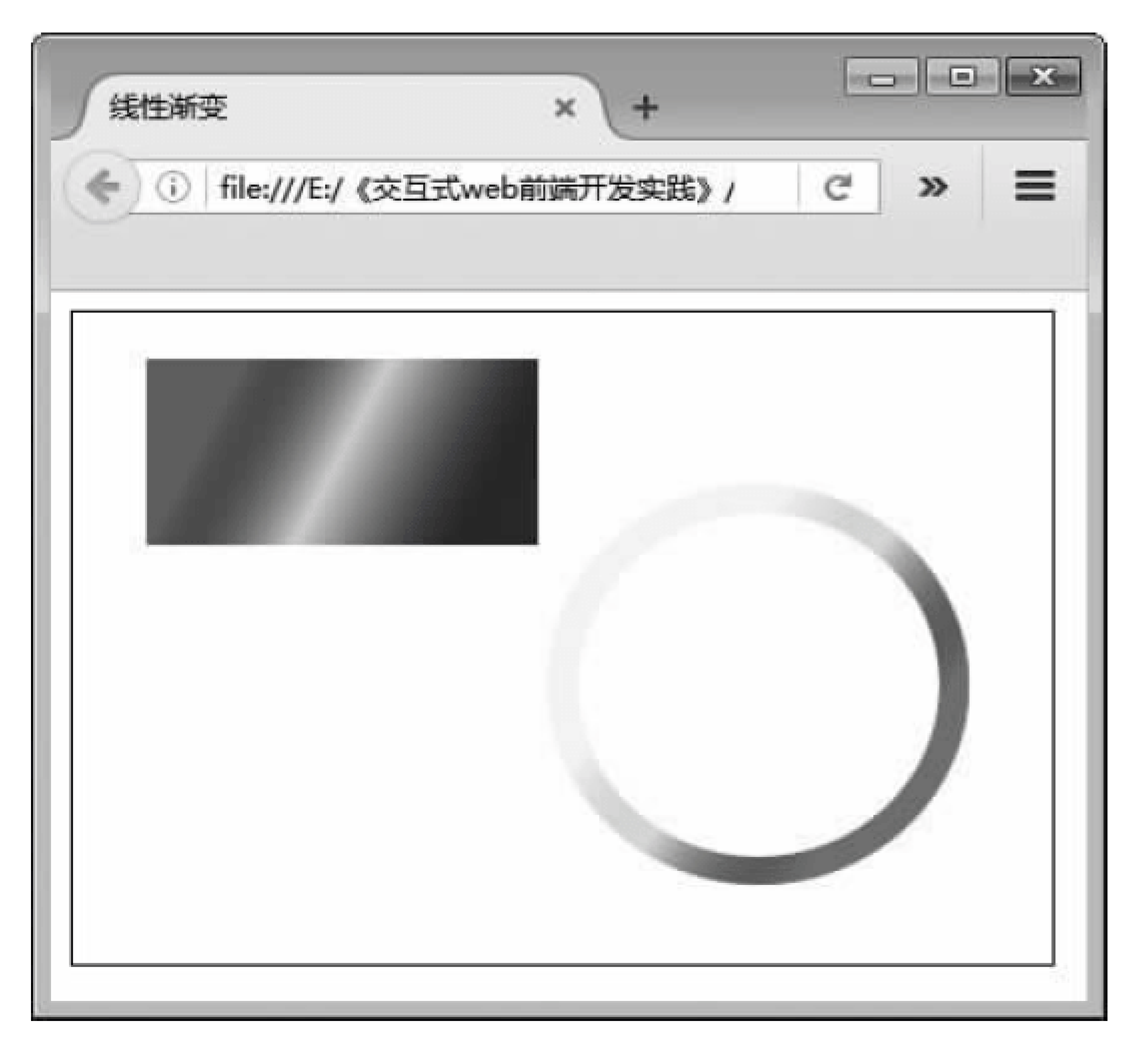


图 2-34　线性渐变

（2）径向渐变

调用 createRadialGradient(sx,sy,sr,ex,ey,er)方法创建径向渐变，与线性渐变的步骤完全类似。其中，参数 sx、sy 为渐变开始时圆的圆心坐标，sr 为渐变开始时圆的半径；参数 ex、ey 为渐变结束时圆的圆心坐标，er 为渐变结束时圆的半径。

【例 2-23】 径向渐变。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>径向渐变</title>
</head>

<body>
    <canvas id="rg" width="400" height="280" style="border:1px solid #000;"></
    canvas>
    <script type="text/javascript">
        var cvs=document.getElementById("rg").getContext("2d");
        cvs.save();
        cvs.translate(30,20);
        var rg=cvs.createRadialGradient(80,40,5,80,40,60);
        rg.addColorStop(0.2,"#f00");
        rg.addColorStop(0.5,"#0f0");
        rg.addColorStop(0.9,"#00f");
        cvs.fillStyle=rg;
        cvs.fillRect(0,0,160,80);
        cvs.restore();
        cvs.translate(280,160);
        cvs.beginPath();
        cvs.arc(0,0,80,0,2*Math.PI);
        cvs.closePath();
        var rg2=cvs.createRadialGradient(0,0,5,0,0,80);
        rg2.addColorStop(0.1,"#ff0");
        rg2.addColorStop(0.4,"#0ff");
        rg2.addColorStop(0.8,"#f0f");
        cvs.fillStyle=rg2;
        cvs.fill();
    </script>
</body>
</html>
```

程序的运行结果如图 2-35 所示。

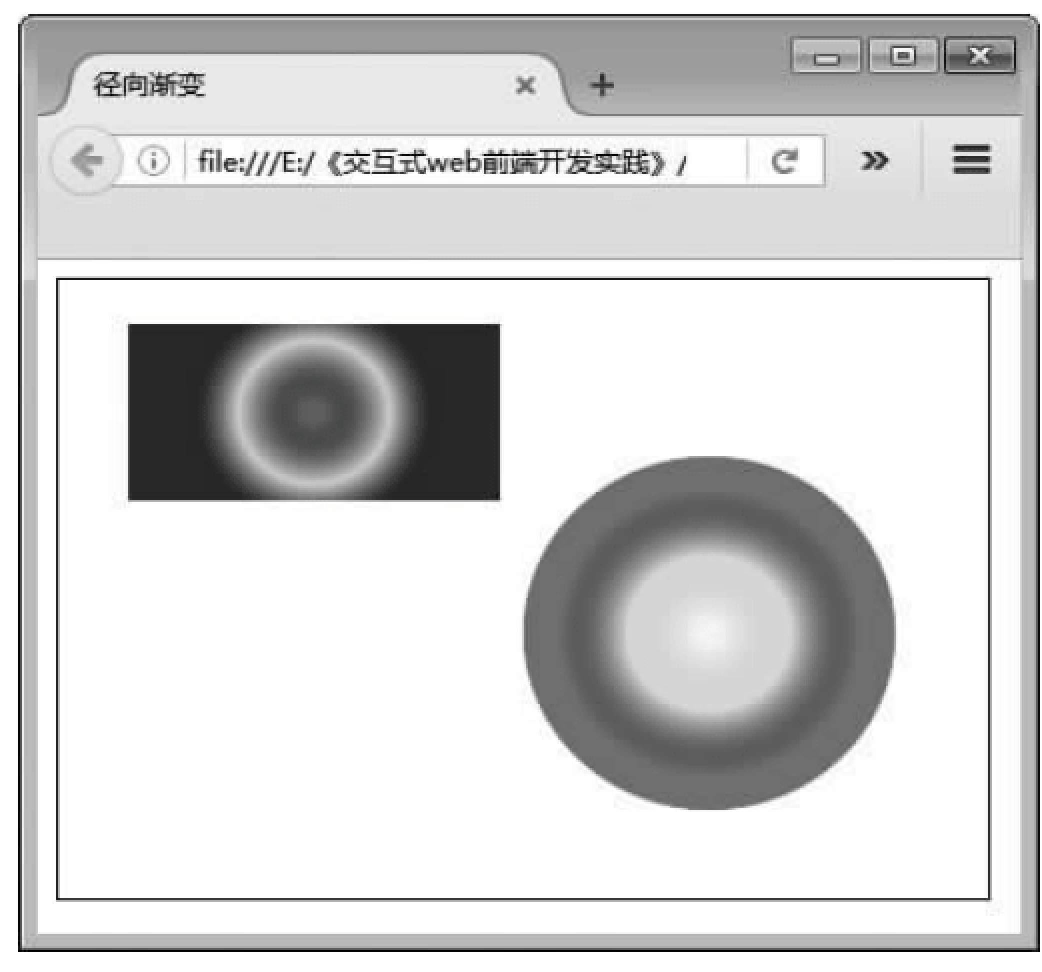


图 2-35　径向渐变

canvas 画布功能是 HTML 5 中新增的一个很重要的功能，这里只介绍了一些基础知识及使用画布绘制图形的常用功能。其余的很多功能，如果读者感兴趣可以再深入研究。

本章小结

本章主要介绍了 HTML 的一些基础知识。通过本章的学习，可以了解 HTML 的语法及编写规范，即如何创建一个 Web 页面。其中包含 HTML 中的各种标签的应用，目前使用的 HTML 的规范，HTML 与 XHTML 的区别与联系。除此之外，简单介绍了目前比较流行的 HTML 5 的应用。HTML 5 新增了很多标签和属性，大大增强了 Web 页面的功能，使 Web 页面的开发更快捷、更高效。

第 3 章　CSS 层叠样式表

3.1　CSS 2 基础

3.1.1　CSS 编写规范

CSS(cascading style sheet)称为层叠样式表，也叫 CSS 样式表或样式表，文件的扩展名为.css，用来设置 Web 页面中各种标签的样式，如设置文字大小、颜色、行高、背景等。所谓“层叠”是指当在 HTML 文件中引用多个样式文件时，浏览器将依据层叠顺序及就近原则进行处理，以避免发生冲突。CSS 样式表是一种控制网页样式，并将样式信息与网页结构分离的一种标记性语言。

1. CSS 基础语法

CSS 样式表由若干条样式规则组成。这些样式规则可以应用到 HTML 的标签中来定义页面的显示效果。样式规则由选择符(selector)、属性(property)及属性值(value)组成，格式如下：

```
selector{property:value;property:value;property:value;…}
```

从以上的格式中可以看出，一条样式规则包含一个选择符和若干属性及对应的属性值。属性及属性值用大括号包含起来；属性与属性值之间用冒号“:”隔开；属性值对之间用分号“;”隔开。如为段落标签<p>设置样式如下：

```
p{font-family:"微软雅黑";font-size:24px;color:#ff0000;}
```

以上样式设置段落文字字体为微软雅黑，字体大小 24 像素，字体颜色为红色。

2. CSS 样式表类型

CSS 样式表有三种类型：行内样式表、内部样式表和外部样式表。

(1) 行内样式表

行内样式表在这三种样式表中是最简单、最直观的一种样式表。它是直接将样式规则写入 HTML 文件中标签的 style 属性中，以属性值的形式存在。如对段落标签设置样式如下：

```
<p style="font-family:'微软雅黑';font-size:24px;color:#ff0000;">段落文字
</p>
```

行内样式表是将样式规则以属性值的形式直接写入标签的 style 属性中,并直接作用于该标签。虽然简单直观,但如果其他的标签或者相同的标签要采用一样的样式,也需要在标签中设置 style 属性,并设置同样的样式规则。这样整个 HTML 页面结构不但显得臃肿,还没有起到页面结构和样式代码进行分离的作用,后期维护也很困难。因此,这种类型的样式表并不常用。

(2) 内部样式表

内部样式表是将样式规则写在<style>和</style>标签之间,并通常将该段样式代码添加到 HTML 页面的头部,即<head>与</head>标签内部。其格式如下:

```
<head>
<meta charset="utf-8">
<title>径向渐变</title>
<style>
    P{
        font-family:"微软雅黑";
        font-size:24px;
        color:#ff0000;
    }
</style>
</head>
```

内部样式表虽然没有完全将页面结构和样式表分离开来,但对于一些简单且某些标签样式需要统一的页面来说,使用内部样式表比使用行内样式表的页面代码结构更优化。

(3) 外部样式表

外部样式表是用户重新创建的一个扩展名为. css 的文件。样式规则按照语法格式直接写入该文件中,不再添加任何标签。

3. 文档中如何引用 CSS 文件

行内样式表和内部样式表都是直接写在 HTML 文档中的,所以不存在引用的问题。但外部样式表是一个独立的文件,与 HTML 文档没有关联性。HTML 文档要使用外部样式表呈现出样式的显示效果,就必须将两个文档关联起来。它们是如何联系的呢? 换句话说,HTML 文档是如何引用 CSS 外部样式表的呢? 有两种方式:链接和导入。下面分别介绍。

(1) 链接外部样式表

链接外部样式表是将外部样式表文件通过<link>标签链接到 HTML 文档的头部即<head>与</head>之间,代码如下:

```
<head>
    <meta charset="utf-8">
    <title>径向渐变</title>
    <link rel="stylesheet" type="text/css" href="*.css" />
</head>
```

其中,属性 rel 表示链接到样式表,值为 stylesheet。属性 type 表示样式表类型为 CSS 样式表。属性 href 指定 CSS 样式表文件的路径,可以使用相对路径或者绝对路径。

链接外部样式表是 CSS 使用中最频繁、最实用的方式，也是目前制作 HTML 页面最常用的一种方式。它最大的优势就是将页面结构 HTML 代码和样式文件 CSS 代码实现了完全的分离。同一个 CSS 文件可以被多个 HTML 文件链接，一个 HTML 文件也可以链接多个 CSS 文件。为了实现相同的样式风格，可以将同一个 CSS 文件链接到网站所有的页面中。如果需要修改样式，只需修改这一个 CSS 文件即可。这使得程序员在进行 Web 前端开发和后期的维护时都十分方便、有效。

(2) 导入外部样式表

导入外部样式表是将外部样式表文件通过使用@import 导入内部样式表<style>与</style>标签中，且放在内部样式表的开始部分。代码如下：

```
<head>
<meta charset="utf-8">
<title>径向渐变</title>
<style>
    @ import "* .css";
    …
</style>
</head>
```

导入外部样式表相当于将外部样式表导入内部样式表中，与链接外部样式表类似，同样实现了结构代码与 CSS 代码的分离。一个 HTML 文件中可以导入多个 CSS 文件，一个 CSS 文件同样可以被多个 HTML 页面导入。

(3) 链接外部样式表与导入外部样式表的区别

链接外部样式表和导入外部样式表从本质上来说都实现了结构代码与样式代码的分离。但它们的语法和运作方式有所不同。

从语法上看，链接外部样式表采用的<link>标签将样式文件链接进了 HTML 文件。导入外部样式表采用的是@import，将样式文件导入了 HTML 文件的内部样式表中。

从运作方式来看，链接外部样式表是 HTML 文件中的标签需要什么样的样式风格时才将外部样式表链接进 HTML 文件。导入外部样式表则是在 HTML 文件初始化时，即浏览器加载 HTML 文件时，就将外部样式表中所有的样式代码导入 HTML 文件的内部样式表中。导入方式相较于链接方式在浏览器加载页面时比较耗时，但显示效果没有任何差别。

在 HTML 文档中引用 CSS 文件不管采用哪种方式，最好是根据需求灵活处理。

3.1.2　CSS 选择符

CSS 选择符也叫 CSS 选择器，HTML 页面中的样式都是通过 CSS 选择器来进行控制的。CSS 选择器可以是 HTML 的结构标签、类、ID 或者是标签的某种状态（如“a：hover”）。因此，可以把 CSS 选择器分为标签选择器、类选择器、ID 选择器和伪类选择器。

1. 标签选择器

标签选择器是使用 HTML 文档中的标签名作为选择符来定义样式规则。如下面的代码定义了部分标签的样式。

```
body,p,a,div,td,th{
    font-size:12px;
    color:#000000;
}
```

通过上面所列举的标签选择器的样式定义，只要页面中出现该标签的地方，就会显示出所定义的样式风格。要设置所有的标签，并不需要将 HTML 中所有的标签名写出来。CSS 提供了通配符选择器“ * ”，它可以代表所有的标签，定义如下：

```
* {margin:0; padding:0;}
```

2. 类选择器

使用标签选择器可以定义页面中所有标签的显示样式，但是如果要对同一种标签在同一个页面中的不同地方使用不同的样式，仅依靠标签选择器是实现不了的。这时就需要类选择器来达到目的。

HTML 中的每一个标签都具有 class 属性，并给该标签定义一个类名，即 class 属性的属性值。这样相同的标签可以具有不同的类名。在设置样式时通过类名来定义不同的样式。

标签的类名作为 CSS 的类选择器时需要在类名前添加符号“. ”，用以定义样式规则，代码格式如下：

```
.black{
    font-size:12px;
    color:#000000;
}
```

或者

```
p.black{
    font-size:12px;
    color:#000000;
}
```

前提是存在结构代码：

```
<p class="black">段落文字</p>
```

类选择器所定义的样式只呈现在具有该类名的标签的显示效果。如果标签中没有该类名，则该标签在页面上的显示效果不具有该类所定义的样式风格。某类选择器所定义的样式还可以应用到其他标签中。如上面定义的“. black”除应用到<p>标签中，还可以应用到<a>标签中，同样有效。但是上面代码中所定义的“p. black”则只能应用在定义了类名为“black”的<p>标签中，不能应用于其他的标签(如：文字)，该标签不具有类“black”所定义的样式效果。

3. ID 选择器

ID 选择器与类选择器类似，是标签中的 id 属性的属性值设置的名称作为选择器来定

义样式规则。ID 名作为选择器定义样式规则时需要在 ID 名前添加符号“#”，样式定义代码格式如下：

```
#black{
    font-size:12px;
    color:#000000;
}
```

或者

```
P#black{
    font-size:12px;
    color:#000000;
}
```

同样需要结构标签中定义 id 属性。

与类选择器所不同的是，一个网页文件中的标签只能使用一次 ID 名，如果其他的标签需要使用，则要另命名一个 ID 名称，即一个 ID 名称在一个页面文件中只能使用一次，不能重复使用。

类选择器与 ID 选择器的区别：类选择器可以在任意数量的标签中使用，而 ID 选择器在页面的标签中只能使用一次；类选择器的优先级比 ID 选择器的优先级低，即当 ID 选择器与类选择器发生冲突时，优先使用 ID 选择器定义的样式。

4. 伪类选择器

伪类选择器所定义的样式规则并不作用在某个具体的标签上，而是作用在标签的状态上。伪类选择器有：:first-child、:focus、:lang、:link、:visitited、:hover、:active 等。其中最常用的是超链接的伪类，即 a:link、a:visitited、a:hover 和 a:active。它们的状态描述见表 3-1。

表 3-1　超链接的伪类

伪类选择器	状态描述	伪类选择器	状态描述
a:link	未访问的超链接	a:hover	鼠标光标停留在超链接上
a:visitited	已访问的超链接	a:active	激活超链接

伪类选择器的应用减少了文档对于类和 ID 的定义，使得文档更简洁。CSS 3 中还新增了一些结构伪类选择器，通过文档结构树来匹配 HTML 中特定的标签。CSS 3 中新增的结构伪类选择器见表 3-2。

表 3-2　结构伪类选择器

选择器	状态描述
E:root	文档的根元素，HTML 文档就是 HTML 元素
E:nth-child(n)	其父元素的第 n 个子元素，第一个元素对应的 n 为 1
E:nth-last-child(n)	其父元素的倒数第 n 个子元素，倒数第一个元素对应的 n 为 1
E:last-child	父元素的最后一个子元素，相当于 nth-last-child(1)

续表

选 择 器	状 态 描 述
E:only-child	父元素下仅有的一个子元素
E:nth-of-type(n)	父元素下使用同种标签的第 n 个子元素，与“:nth-child(n)”类似
E:nth-last-of-type(n)	父元素下使用同种标签的倒数第 n 个子元素，与“:nth-last-child(n)”类似
E:first-of-type	父元素下使用同种标签的第一个子元素，相当于“:nth-of-type(1)”
E:last-of-type	父元素下使用同种标签的最后一个子元素，相当于“:nth-last-of-type(1)”
E:only-of-type	父元素下使用同种标签的唯一一个子元素
E:empty	一个不包含任何子元素的标签。注意：文本节点也被看作子元素

伪类选择器在设置页面中某些特定样式时非常有用且高效，可以避免在页面结构中使用大量的类或 ID 来完成。如页面中的某一表格奇数行与偶数行的样式不一致时，采用伪类选择器就可以避免在结构代码中添加大量的类来完成。

3.1.3 文本样式

文本在 HTML 中很重要，可以说是网页的核心内容。默认情况下，文本是黑色的，字体大小根据浏览器不同而有所不同，一般情况下为 16 像素，字体以宋体或者微软雅黑为主。在网页中，文本的颜色、间距、行距、大小、字体等效果多种多样，必须根据设计需求进行排版，才能使页面看起来主次分明。文本样式包括字体属性和段落属性。

1. 字体属性

虽说文本是网页中最主要的内容，如果排列得杂乱无章，会让人觉得枯燥乏味，但如果版式美观大方，并结合网页的其他元素合理进行样式设置，会让人眼前一亮，让人流连忘返，使网页内容给浏览者留下深刻的印象。字体属性及描述见表 3-3。

表 3-3 字体属性及描述

属 性	描 述
font-family	指定文字字体的类型，如宋体、黑体、Time New Roman 等
font-size	指定文字的大小，通常用像素值表示
font-style	定义字体的风格，属性值有 normal、italic、oblique、inherit
font-weight	设置字体的粗细，属性值有 normal、bold、bolder、lighter、100～900，数值越大，加粗的程度越高
font-variant	设置文本中所有的小写字母均会被转换为大写。属性值有 normal(默认值)、small-caps(显示小型大写字母)、inherit(继承父元素的该属性)
color	设置字体颜色，属性值可以是颜色名称、十六进制和 RGB 代码等

以上列出的这些属性都是字体样式设置中常用的属性。但这些属性都是将字体的各种样式分开进行设置的。CSS 还提供了字体样式设置的复合属性 font 属性。font 属性可以一次性设置多个属性值来定义字体样式，其格式如下：

```
{font:font-style font-variant font-weight font-size font-family}
```

字体复合属性 font 中属性值排列顺序为 font-style、font-variant、font-weight、font-size 和 font-family，各属性值之间用空格隔开。如果 font-family 属性值有多个，则使用逗号“，”分隔。属性 font 的属性值顺序是不可以随意调换的，但其中的属性值可以省略不写，但不可以前后调换顺序。值得注意的是，font-size 和 font-family 这两个属性值不可省略，否则 font 属性所设置的样式规则可能会被忽略。

2. 段落属性

段落的应用在网页中也是常见的，是文章的组成部分。段落的放置与显示直接影响到了页面的布局及风格。CSS 提供了一些文本属性对段落文字进行控制，见表 3-4。

表 3-4　段落属性及描述

属　　性	描　　述
word-spacing	单词之间的间隔，只针对单词有效，对中文词语无效
letter-spacing	字符间距，对中英文均有效
text-decoration	设置文字的修饰，属性值有 underline、overline、line-through、blink、none
text-transform	文本字体的大小写转换，属性值有 none(不转换)、capitalize(每个单词的第一个字母转换成大写，其余不转换)、uppercase(转换成大写)、lowercase(转换成小写)
text-align	文本水平对齐方式，只能用于文本块，不能独立应用于图像标签＜img＞
text-indent	文字缩进。段落文字通常首行会缩进两个字符
line-height	设置行间距，即行高
white-space	设置文本间空白的处理方式。属性值有 normal(默认值，忽略不处理)、pre(保留空白)、nowrap(不换行)、pre-wrap(保留空白，能正常换行)、pre-line(合并空白，保留换行符)、inherit(继承父元素的该属性)
direction	用于指定文本的排列方向。属性值为 ltr 时，文本的方向从左到右，这也是文本排列的默认值；属性值为 rtl 时，文本的方向从右到左

段落属性也是文本样式中的一部分，主要设置段落文字的样式。这些属性的应用主要完成段落文字的排版及显示效果，使整个网页看起来美观大方，能够吸引更多的访问者。

3.1.4　背景边框样式

打开网页，首先映入眼帘的便是该网页的颜色基调及样式风格。不同类型的网页有不同的设计风格。背景是网页样式风格中的一个重要因素。本节将介绍背景和边框的样式设置。

1. 背景属性

很多网站主题都是以网页的背景为样式风格的。背景的设置可以提高网页的视觉性，吸引不少访问者。CSS 样式表的强大表现功能对背景的样式设置提供了一些属性，见表 3-5。

(1) background-color 属性

background-color 属性设置页面的背景颜色。属性的值可以是英文的颜色名称、十六进制编码、RGB 值、HSL 值、HSLA 值和 GRBA 值。background-color 属性可以设置整个页面的背景颜色，也可以设置指定的某个 HTML 标签区域的背景颜色。

表 3-5　背景属性及描述

属　性	描　述
background-color	设置背景颜色
background-image	设置背景图片
background-repeat	设置背景图片是否重复
background-attachment	设置背景图片的显示方式
background-position	设置背景图片的位置
background	复合属性,可以一次性指定上面的所有属性

(2) background-image 属性

background-image 属性可以将图像设置为页面的背景图片。值得注意的是没有任何方法可以控制背景图片在背景设置中的宽度和高度,即背景图片有多大的尺寸,在背景区域中就可以显示多大的尺寸,如果背景区域大于背景图片的大小,背景图片则默认平铺背景区域;如果背景区域小于背景图片的大小,背景图片则舍弃一部分不显示,只显示背景区域提供的大小。因此,如果想要将背景图片在背景设置中控制预期的宽度和高度,则需要预先处理好该图片,通过其他图片处理工具预先将图片处理好。另外,背景图片是作为区域的背景显示的,因此不存在像在页面中插入图片那样的 alt 属性,当图片无法正常显示时,alt 属性提供替换的文本描述。因此,背景图片不能用于传达任何没有在页面上使用文本描述的信息。

background-image 属性和 background-color 属性可以同时设置在同一个 HTML 标签区域中,但显示的时候,背景图片会显示在背景颜色的上层,即背景图片会遮挡背景颜色的显示。

(3) background-repeat 属性

background-repeat 属性用于设置背景图片是否重复平铺。背景图片的重复平铺方式有两种:水平重复平铺和垂直重复平铺。background-repeat 属性及描述如表 3-6 所示。

表 3-6　background-repeat 属性及描述

属　性	描　述
repeat	背景图片既水平平铺又垂直平铺,即铺满整个标签区域
repeat-x	背景图片水平方向平铺
repeat-y	背景图片垂直方向平铺
no-repeat	背景图片不重复平铺

background-repeat 属性重复平铺图片是从标签区域的左上角开始按水平方向和垂直方向重复铺满整个区域的。

(4) background-attachment 属性

background-attachment 属性用于设置背景图片的显示方式。当一个页面过大,而背景图片又不能完全覆盖整个区域的时候,浏览页面会出现看不到背景图片;或是开始时可以看见,但滚动页面就看不见背景图片的情况,即背景图片不能随页面的滚动而显示。background-attachment 属性就是用来设置背景图片是否随页面滚动而显示的。background-attachment 属性及描述如表 3-7 所示。

表 3-7　background-attachment 属性及描述

属　性	描　述
scroll	默认值。当页面滚动时背景图片随页面一起滚动
fixed	背景图片固定在浏览器中显示页面的可见区域

如果不设置 background-attachment 属性，背景图片固定在页面中的某个区域。这里要区分设置了 background-attachment 属性值为 fixed 时的情况。当 background-attachment 属性值为 fixed 时，背景图片固定的位置并不是相对于页面的，而是相对于页面在浏览器中的可见区域。而不设置该属性时，背景图片固定的位置是相对于页面的，而不是在浏览器的可见区域。

（5）background-position 属性

background-position 属性用于设置背景图片在页面中的具体位置。它的属性值可以是：绝对定义的位置、百分比定义的位置、垂直方向值和水平方向值。background-position 属性及描述如表 3-8 所示。

表 3-8　background-position 属性及描述

属　性	描　述
length	设置位置的水平和垂直方向的具体距离长度，单位可以为 cm、mm、px 等
percentage	设置标签区域宽度和高度的百分比定义的背景图片所在位置
top	设置背景图片在垂直方向上顶部对齐
bottom	设置背景图片在垂直方向上底部对齐
center	设置背景图片在水平和垂直方向上居中对齐
left	设置背景图片在水平方向上左对齐
right	设置背景图片在水平方向上右对齐

background-position 属性中背景图片的位置设置可以同时指定水平方向值和垂直方向值，两个属性值用空格隔开。水平方向值和垂直方向值除了可以用方向定位词（left、top、center、bottom 和 right）外，还可以用具体的距离数值（length）和百分比（percentage）。

（6）background 复合属性

background 属性是一个复合属性，可以一次性设置多个背景属性，即可以将 background-color 属性、background-image 属性、background-repeat 属性、background-attachment 属性和 background-position 属性一起设置，中间用空格符号隔开。这些背景属性值可以按照任何顺序排列，不分先后。属性值的排列顺序不像是 font 复合属性值必须按照一定的顺序排列。

2．边框属性

所谓边框，就是将内容和间距包含在内的边线，类似于表格的外边线。边框可以从三个方面来描述：样式、线宽和颜色。边框属性及描述如表 3-9 所示。

表 3-9　边框属性及描述

属　　性	描　　述
border-style	设置边框的样式
border-width	设置边框的线宽
border-color	设置边框的颜色
border	复合属性,可以指定上面所有的边框属性

（1）border-style 属性

border-style 属性用来设置边框的样式风格,即边框线是采用实线、虚线或是点线以及双实线等。border-style 属性及描述如表 3-10 所示。

表 3-10　border-style 属性及描述

属　性	描　　述
none	无边框
dotted	边框样式为点线
dashed	边框样式为虚线
solid	边框样式为实线
double	边框样式为双实线
groove	边框样式如同雕刻的 3D 凹槽式边框
ridge	3D 凸槽、脊线式边框,与 groove 边框看上去相反
inset	3D 凹入内嵌式边框,看上去如同嵌入在页面中
outset	3D 凸出突出式边框,看上去如同位于画布之外

一个边框还可以分为四个方向上的边框线,可以分别进行边框样式设置。通过 border-top-style 属性设置上边框线的样式;border-right-style 属性设置右边框线的样式;border-bottom-style 属性设置下边框线的样式;border-left-style 属性设置左边框的样式。另外,这四个方向上的边框样式还可以同时设置,其语法格式如下:

```
border-style:dashed solid dotted double;
```

将四个边框样式设置在一个 border-style 属性中,中间用空格隔开。按照顺时针的方向分别定义上边框、右边框、下边框和左边框的样式。

（2）border-width 属性

border-width 属性用于设置边框的线宽,通常使用像素值来指定宽度。border-width 属性的值不能使用百分比,除了使用具体的数值外,还可以使用 thin、medium 和 thick 这三种属性值。medium 是默认值,thin 比 medium 细,thick 比 medium 粗,但显示的实际宽度还取决于所使用的浏览器。

border-width 属性其实也是 border-top-width、border-right-width、border-bottom-width 和 border-left-width 的综合属性,分别设置上、右、下、左边框线的宽度。如果四个方向上的框线宽度都设置在一个 border-width 属性值中,则按顺时针的方向对上、右、下、左边框分别设置框线的宽度。

（3）border-color 属性

border-color 属性用于设置边框的颜色。在没有设置边框样式的情况下,默认的边框颜

色基于页面元素的 color 值。同 border-style 和 border-width 属性一样，border-color 属性可以为边框设置一种颜色，也可以同时设置四个边框的颜色，按顺时针方向分别设置上、右、下、左边框的颜色。也可以单独设置某一边上的边框。border-top-color 属性用于设置上边框颜色；border-bottom-color 属性用于设置下边框颜色；border-left-color 属性用于设置左边框颜色；border-right-color 属性用于设置右边框颜色。

（4）border 复合属性

border 属性是一个复合属性，集合了上面所介绍的三种属性。其格式如下：

```
border:border-style border-width border-color;
```

border 属性将边框框线样式、框线宽度、框线颜色集合在一起进行设置，属性值之间用空格隔开。三个属性值的顺序可以自由调换。

3.1.5　列表样式

CSS 提供了用于控制列表样式的属性，如表 3-11 所示。

表 3-11　列表属性及描述

属　性	描　述
list-style-type	设置列表项目符号或编号的形状或外观显示，即列表项目符号类型
list-style-position	设置列表项目符号或编号的位置显示
list-style-image	指定一张图片作为列表的项目符号或编号
list-style	复合属性，可以同时设置列表项目符号或编号的类型和位置

1. list-style-type 属性

list-style-type 属性用于设置列表项目符号或编号的样式。对于无序列表的项目符号样式设置的属性值说明如表 3-12 所示。

表 3-12　无序列表的 list-style-type 属性

属　性	描　述	属　性	描　述
none	没有项目符号或编号	circle	空心圆
disc(默认值)	实心圆	square	方块

无序列表使用的是项目符号，有序列表使用的是编号。list-style-type 属性对于有序列表的编号样式设置的属性值的说明如表 3-13 所示。

表 3-13　有序列表的 list-style-type 属性

属　性	描　述
decimal	数字，如 1、2、3
decimal-leading-zero	前面添加 0 的数字，如 01、02、03
lower-alpha	小写字母，如 a、b、c
upper-alpha	大写字母，如 A、B、C
lower-roman	小写罗马数字，如 i、ii、iii、iv、v

list-style-type 属性可以用于无序列表<ul>标签，也可以用于有序列表<ol>标签，还可以用于列表项<li>标签。

2．list-style-position 属性

list-style-position 属性用于设置列表项目符号或编号的位置显示。该属性具有两个属性值：inside 和 outside，如表 3-14 所示。

表 3-14　list-style-position 属性及描述

属　性	描　　述
inside	项目符号或编号在文本区域内左侧(缩进的)
outside	项目符号或编号在文本区域外左侧(默认值)

【例 3-1】 list-style-position 属性的两个属性值的对比。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>list-style-position 属性</title>
<style>
    ul{list-style-position:inside;}
    ol{list-style-position:outside;}
</style>
</head>

<body>
<ul>
    <li>list-style-position 属性值为 inside 时，项目符号或编号在文本区域内左侧(缩
    进的)。</li>
</ul>
<ol>
    <li>list-style-position 属性值为 outside 时，项目符号或编号在文本区域外左侧(默
    认值)。</li>
</ol>
</body>
</html>
```

程序的运行结果如图 3-1 所示。



图 3-1　list-style-position 的两个属性值的对比显示效果

由此可见，list-style-position 属性所定义的列表文字都是缩进的，当属性值为 inside 时，项目符号或编号所在的位置在列表文本区域内；当属性值为 outside 时，项目符号或编号所在的位置在列表文本区域外。

3. list-style-image 属性

list-style-image 属性是指定一张图片作为列表的项目符号或编号。属性值是图片的源路径，即用 url()方法引入使用图片的路径。如果使用的是嵌套的列表，该属性将继承父元素的 list-style-image 属性。如果不想继承该属性，则将该属性值设置为 none 即可。

4. list-style 属性

list-style 属性是列表的复合属性，可以一次性地设置列表项目符号或编号的样式类型及位置，属性值之间用空格隔开。属性值无先后顺序，可以自由调换。

3.1.6　其他样式

还有几个常用的样式属性：cursor 属性、display 属性、visibility 属性、overflow 属性和 opacity 属性。下面分别进行介绍。

1. cursor 属性

cursor 属性是一种鼠标属性，可以指定鼠标光标的显示类型。cursor 属性及描述如表 3-15 所示。

表 3-15　cursor 属性及描述

属　性	描　　述
auto	默认值。浏览器设置的光标
crosshair	十字线光标
default	通常是一个箭头
pointer	手指形光标，通常用于指示链接的指针
move	握着的手形光标，如正在拖放某个对象时的光标显示
e-resize	此光标指示矩形框的边缘可被向右(东)移动
ne-resize	此光标指示矩形框的边缘可被向上及向右移动(北/东)
nw-resize	此光标指示矩形框的边缘可被向上及向左移动(北/西)
n-resize	此光标指示矩形框的边缘可被向上(北)移动
se-resize	此光标指示矩形框的边缘可被向下及向右移动(南/东)
sw-resize	此光标指示矩形框的边缘可被向下及向左移动(南/西)
s-resize	此光标指示矩形框的边缘可被向下移动(南)
w-resize	此光标指示矩形框的边缘可被向左移动(西)
text	类似于竖线，指示文字的内容
wait	指示忙，正在等待，光标通常是一只沙漏的显示
help	指示可用的帮助，光标通常是一个问号或气球
url	自定义光标图像文件的源地址，通常在此列表的末端始终定义一种普通的光标，以防没有由 URL 定义的可用光标

其中用得最多的就是 pointer 属性值，通常把一些不是手指形光标设置为可以指示链接的手指形光标指针，提示用户该处可以进行点击操作。

2. display 属性

display 属性是一种显示属性，可以定义元素的显示类型。display 属性及描述如表 3-16 所示。

表 3-16 display 属性及描述

属　性	描　　述
none	此元素不会被显示
block	此元素将显示为块级元素，此元素前后会带有换行符
inline	此元素会被显示为行内元素，元素前后没有换行符
inline-block	行内块元素（CSS 2.1 中新增的）

属性值 block 是以块级元素的方式显示，inline 是以内联元素的方式显示，none 是不显示，inline-block 则是鉴于行内元素和块级元素之间的一种显示效果。

什么是块级元素？什么是行内元素？

所谓块级元素，就是这类元素应用在 HTML 页面中，显示效果会成为独立的一行，元素前后自带换行符。设置该元素的宽度 width 属性、高度 height 属性以及内外边距属性是有效的，能够显示出设置尺寸大小的元素区域。

所谓行内元素，又称内联元素，这类元素应用在 HTML 页面中，显示效果是紧跟着前面元素的后面显示，不会换行。设置该元素的宽度 width 属性、高度 height 属性以及内外边距属性是无效的，只能显示元素内容自身所占的区域尺寸大小，不会显示出设置尺寸的区域。

display 属性可以将块级元素和行内元素进行相互转换，即将块级元素 display 属性值设置为 inline，则该元素转换成了行内元素，具有行内元素的特性；将行内元素 display 属性值设置为 block，则该元素转换成了块级元素，具有块级元素的特性。

行内元素是无法定义宽高尺寸。对行内元素设置高度、宽度、内外边距等属性都是无效的，它的宽度、高度都是由内容本身决定的。当将一个内联元素（如标签<a>）改为块元素后，该元素具有块级元素的特性，会单独占据一行，其他跟在它的后面的元素会被迫换行，转到下一行，也可以通过设置高度、宽度、内外边距等属性来调整这个元素区域的尺寸大小。

对于 display 属性值设置为 inline-block 的元素，自身的显示效果是块级元素，即设置宽度、高度及内外边距有效，相对于前后周围的元素来说它却是行内元素，具有行内元素的显示效果，即该元素前后不会换行，紧挨着前面元素内容的后面显示该元素的内容。

display 属性可以设置元素是否显示出来。属性值设置为 none 则不显示元素，属性值设置为 block 则将元素显示出来并具有块级元素的特性。

3. visibility 属性

visibility 属性设置元素是否可见。visibility 属性及描述如表 3-17 所示。

表 3-17 visibility 属性及描述

属　性	描　　述
visible	默认值，将元素设置为可见
hidden	将元素设置为不可见，即隐藏

当将 visibility 属性的值设置为 visible 时，元素在浏览器中正常显示；当将 visibility 属性的值设置为 hidden 时，元素在浏览器中隐藏内容，但是该元素不影响页面的布局，即这时看不见元素中的内容，但在浏览器的显示位置会看到元素所占用的空白区域。也就是说 visibility 属性设置为 hidden 的元素虽然在页面中不显示出来，但它会占据一定的位置区域，不会影响页面的布局及其他元素内容的显示。

这里要注意区分 display 属性设置隐藏的显示效果。当 display 属性的值设置为 none 时，该元素被完全隐藏，不占据页面空间。该元素所在页面中的位置会被后面的元素内容填补占据。而 visibility 属性设置为 hidden 时，虽然看不见内容，却可以看见该内容在页面中所占据的空间位置，其他元素内容不会来填补该区域。

4. overflow 属性

overflow 属性规定当内容溢出时发生的事情。overflow 属性及描述如表 3-18 所示。

表 3-18　overflow 属性及描述

属　性	描　　述
visible	默认值。内容不会被修剪，会呈现在元素框之外
hidden	内容会被隐藏，并且其余内容是不可见的
scroll	内容会被修剪，但是浏览器会显示滚动条以便查看其余的内容
auto	根据内容的多少，自动决定是否修剪，并通过滚动条查看其余的内容

扩展用法：overflow-x 属性值和 overflow-y 属性值可以根据水平方向和垂直方向单独规定溢出元素的显示方式。

5. opacity 属性

opacity 属性是用来设置透明度的。当我们需要对某些元素表现为半透明效果时，需要使用 opacity 属性。语法如下：

```
opacity:0.5;
```

opacity 属性值的取值范围为 0～1，1 表示不透明，0 表示完全透明。高版本浏览器对 opacity 已经有了很好的支持。

3.2　CSS 3 基础

3.2.1　CSS 3 新增特性

1. 背景属性

在 CSS 3 中新增了一些用来控制背景图片在区域中的显示样式属性。这些功能在 CSS 2 中是无法控制实现的。新增的背景属性如表 3-19 所示。

表 3-19 CSS 3 新增的背景属性及描述

属 性	描 述
background-size	设置背景图片的大小
background-origin	设置背景的显示区域
background-clip	设置背景图像的裁剪区域

（1）background-size 属性

background-size 属性用于设置背景图片的大小。在前面章节介绍背景属性时提到，背景图片的大小是不可控制的。如果要使背景图片的尺寸适应标签区域的大小，要么将图片预先处理好，要么通过平铺或舍弃一部分来达到目的。在 CSS 3 中，通过 background-size 属性可以设置背景图片以任何尺寸显示，而不是以重复平铺或舍弃一部分图片区域来适应标签区域的尺寸。background-size 属性及描述如表 3-20 所示。

表 3-20 background-size 属性及描述

属 性	描 述
length	设置背景图片的具体尺寸数值，单位可以为 cm、mm、px 等，不可为负值
percentage	设置背景图片的百分比尺寸，不可为负值
cover	按背景图片自身的宽高比例缩放到正好完全覆盖所定义的标签区域
contain	保持背景图片自身的宽高比例，将图片缩放到正好适应所定义的标签区域的宽度或高度

当 background-size 属性值设置为百分比时，背景图片的尺寸大小由所定义的标签区域的宽度、高度以及 background-origin 属性的位置决定。background-size 属性可以设置 2 个属性值，中间用空格隔开。第 1 个值表示设置的背景图片的宽度，第 2 个值表示设置的背景图片的高度。如果只设置了 1 个值，则第 2 个值默认为 auto，即高度根据宽度的设置及自身的宽高比例而定。

（2）background-origin 属性

background-origin 属性指定背景图片的定位方式。background-position 属性总是以标签区域左上角为坐标原点进行的定位。而 background-origin 属性可以改变这种定位方式。background-origin 属性及描述如表 3-21 所示。

表 3-21 background-origin 属性及描述

属 性	描 述
border	从 border 区域开始显示背景图片
padding	从 padding 区域开始显示背景图片
content	从 content 区域开始显示背景图片

（3）background-clip 属性

background-clip 属性用来设置背景图片的裁剪区域。background-clip 属性及描述如表 3-22 所示。

表 3-22 background-clip 属性及描述

属 性	描 述
border	从 border 区域开始显示背景图片
padding	从 padding 区域开始显示背景图片
content	从 content 区域开始显示背景图片
no-clip	从边框区域外裁剪背景

background-clip 属性与 background-origin 属性有些类似。也就是说，background-clip 属性用来判断背景是否包含边框区域，而 background-origin 属性用来决定 background-position 属性定位的参考位置。

（4）多背景图片

CSS 3 允许同时指定多个背景图片，这些背景图片会依次覆盖在页面元素的背景上。CSS 3 并没有为多背景图片提供额外的属性，多背景图片依然是通过 background-image、background-repeat、background-position、background-size 等属性来控制的，格式如下：

```
background-image:url(image/snow.gif),url(image/face.gif),url(image/sky.gif);
background-repeat:repeat, repeat-x, repeat-y;
background-position:left top, center top, left top;
```

CSS 3 允许在一个属性中指定多个属性值，中间用逗号隔开。

2. 边框属性

在 CSS 3 中，新增了一些对于边框样式设置的属性，如圆角边框、图片边框等。这在 CSS 2 中都是会花费大量的时间和精力才能实现的。但在 CSS 3 中能轻松地实现。

（1）圆角边框

在 CSS 3 中，使用 border-radius 属性来设置边框的圆角显示效果，格式如下：

```
border-radius:length;
```

属性值用一个长度数值来表示设置圆角的半径，半径越大，圆角的程度越大。属性值也可以设置为 none，表示无圆角，即边框是直角边框。属性值不能是一个负值。

border-radius 属性的值可以设置 1 个值，也可以设置 2 个值，2 个值中间用斜杠“/”符号隔开。当设置为 1 个值时，表示边框的 4 个圆角半径相同，且每个圆角都是 1/4 个圆形。当设置为 2 个值时，第 1 个值表示圆角的水平半径，第 2 个值表示圆角的垂直半径，即每个圆角并不是 1/4 个圆形。第 2 个值与第 1 个值相同时，就和只设置 1 个值时呈现一样的圆角显示效果。

【例 3-2】 指定两个圆角半径。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
```

```
<title>指定两个圆角半径</title>
<style>
    .rec01{
        width:200px;
        height:80px;
        border:15px solid #F00;
        border-radius:15px;
    }
    .rec02{
        width:200px;
        height:80px;
        border:15px solid #F00;
        border-radius:15px/50px;
    }
</style>
</head>
<body>
    <p class="rec01">这是一个水平半径和垂直半径相同的圆角边框</p>
    <p class="rec02">这是一个水平半径和垂直半径不同的圆角边框</p>
</body>
</html>
```

程序的运行结果如图 3-2 所示。



图 3-2　具有两个半径的圆角边框的显示效果

由此可见，第一个圆角边框的半径设置了一个属性值，每个圆角是一个圆形的 1/4；第二个圆角边框设置了两个属性值，每一个圆角实际上是椭圆的一部分。

border-radius 属性的值还可以为其赋予一组值。这一组值之间的间隔用空格隔开，可以包含 2～4 个属性值。这里要注意区分用斜杠“/”符号隔开的 2 个属性值的情况：用斜杠“/”符号隔开的表示圆角的水平和垂直半径。而这一组值中包含的 2 个属性值的情况是用

空格隔开，且这一组属性值是不能使用斜杠“/”符号定义圆角的水平和垂直半径。这一组属性值的赋值情况如表 3-23 所示。

表 3-23　border-radius 属性赋值个数描述

属性值个数	描　述
2	第一个值表示左上角、右下角的半径，第二个值表示右上角、左下角的半径
3	第一个值表示左上角的，第二个值表示右上角和左下角的，第三个值表示右下角的
4	第一个值表示左上角的，第二个值表示右上角的，第三个值表示右下角的，第四个值表示左下角的

【例 3-3】　定义一组值的圆角半径。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>定义一组值的圆角半径</title>
<style>
    .box1{
        width:200px;
        height:80px;
        border:15px solid #F30;
        border-radius:10px 50px;
    }
    .box2{
        width:200px;
        height:80px;
        border:15px solid #F30;
        border-radius:10px 50px 70px;
    }
    .box3{
        width:200px;
        height:80px;
        border:15px solid #F30;
        border-radius:10px 30px 50px 70px;
    }
</style>
</head>
<body>
    <div class="box1"></div><br>
    <div class="box2"></div><br>
    <div class="box3"></div>
</body>
</html>
```

程序的运行结果如图 3-3 所示。

由此可见，border-radius 属性还有一些衍生属性，如表 3-24 所示。

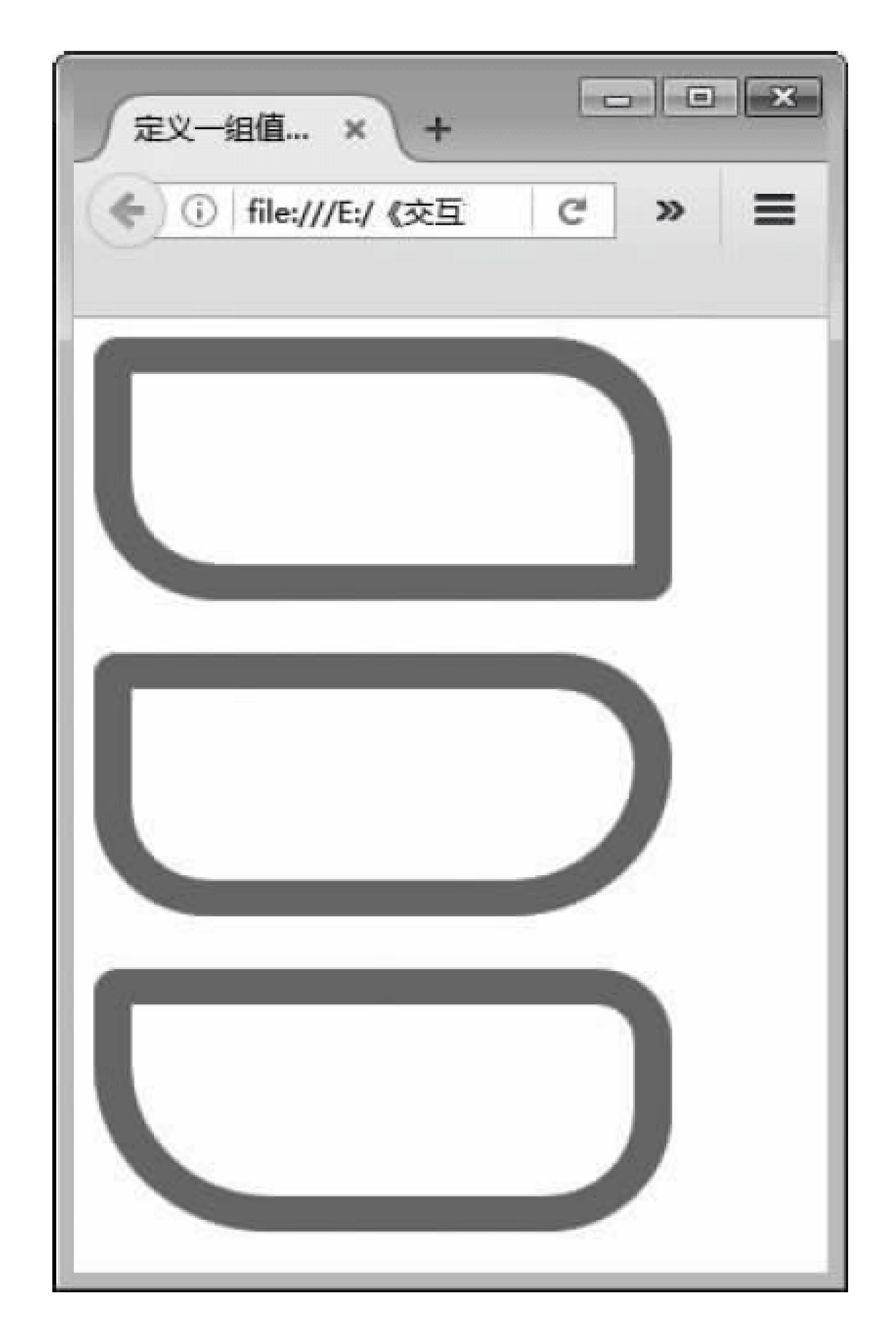


图 3-3　具有一组属性值的圆角边框的对比显示效果

表 3-24　border-radius 的衍生属性及描述

属　　性	描　　述	属　　性	描　　述
border-top-left-radius	定义左上角圆角	border-bottom-left-radius	定义左下角圆角
border-top-right-radius	定义右上角圆角	border-bottom-right-radius	定义右下角圆角

（2）渐变颜色边框

CSS 3 提供了一些渐变颜色的边框属性，即 border-top-colors 属性、border-bottom-colors 属性、border-left-colors 属性和 border-right-colors 属性，设置边框的宽度为 *N*px，就可以设置 *N* 种颜色，每种颜色显示 1px 的宽度。如果设置的颜色数小于边框的宽度，最后一个颜色覆盖剩下的所有宽度。这里的渐变颜色边框属性不能简写成 border-colors。

渐变颜色边框属性目前只有 Firefox 浏览器支持，而且使用时必须在前面加-moz-前缀。其语法格式如下：

```
-moz-border-top-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd #eee;
```

【例 3-4】　定义渐变颜色边框。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>渐变颜色边框</title>
<style>
    .box{
```

```
        width:200px;
        height:80px;
        border-width:15px;
        border-style:solid;
        -moz-border-top-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd #eee;
        -moz-border-bottom-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd
        #eee;
        -moz-border-left-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd #eee;
        -moz-border-right-colors:#555 #666 #777 #888 #999 #aaa #bbb #ccc #ddd #eee;
    }
</style>
</head>
<body>
    <div class="box"></div>
</body>
</html>
```

程序的运行结果如图 3-4 所示。



图 3-4　渐变颜色边框的显示效果

(3) 图片边框

在 CSS 3 中新增了一个 border-image 属性，用来控制边框图片。该属性与 background-image 属性有些类似，但它的作用更强大一些。

border-image 属性有五大属性值，分别是图片地址、图片切割、图片宽度、图片外凸和图片重复。这就形成了 border-image 的衍生属性，见表 3-25。

表 3-25　border-image 的衍生属性及描述

属　性	描　述
border-image-source	指定边框图片的源地址，即图片的 URL
border-image-slice	确定如何裁剪边框图片
border-image-width	指定边框图片的尺寸，实际上浏览器还是习惯使用 border-width 属性来实现
border-image-outset	指定边框图片向边框外延伸的距离
border-image-repeat	指定边框图片的重复性

border-image 属性是一个复合属性，可以将它衍生出来的属性值一次性一起设置，属性值之间用空格隔开即可。

border-image-source 属性是指定用来设置边框图片的源地址，通过使用 url()调用图片，地址可以是相对地址，也可以是绝对地址。属性值还可以设置为 none，即不使用图片边框。

border-image-slice 属性是用于控制如何裁剪用来设置边框的图片。属性值可指定 1～4 个数值或百分比，将图片按照一定的方向切割成 9 个区域，即 4 个角、4 条边和中间部分。其中的 4 个角和 4 条边用来设置边框，中间部分会被舍弃且不显示出来。如果需要将中间部分显示出来，在 border-image-slice 属性值的后面添加 fill 后缀，即在属性值的后面添加 fill 标识，与前面的值之间用空格隔开(如 border-image-slice：30 fill)。其实，把中间部分显示出来就跟使用图片作为背景有些类似了，但也有区别：图片边框是将切割下来的图片分别去适应边框相对应区域的尺寸大小。注意：这里设置属性值为数值时，不能添加单位，且不允许是负值。如图 3-5 所示，展示了 border-image-slice 属性裁剪图片区域的示意图。

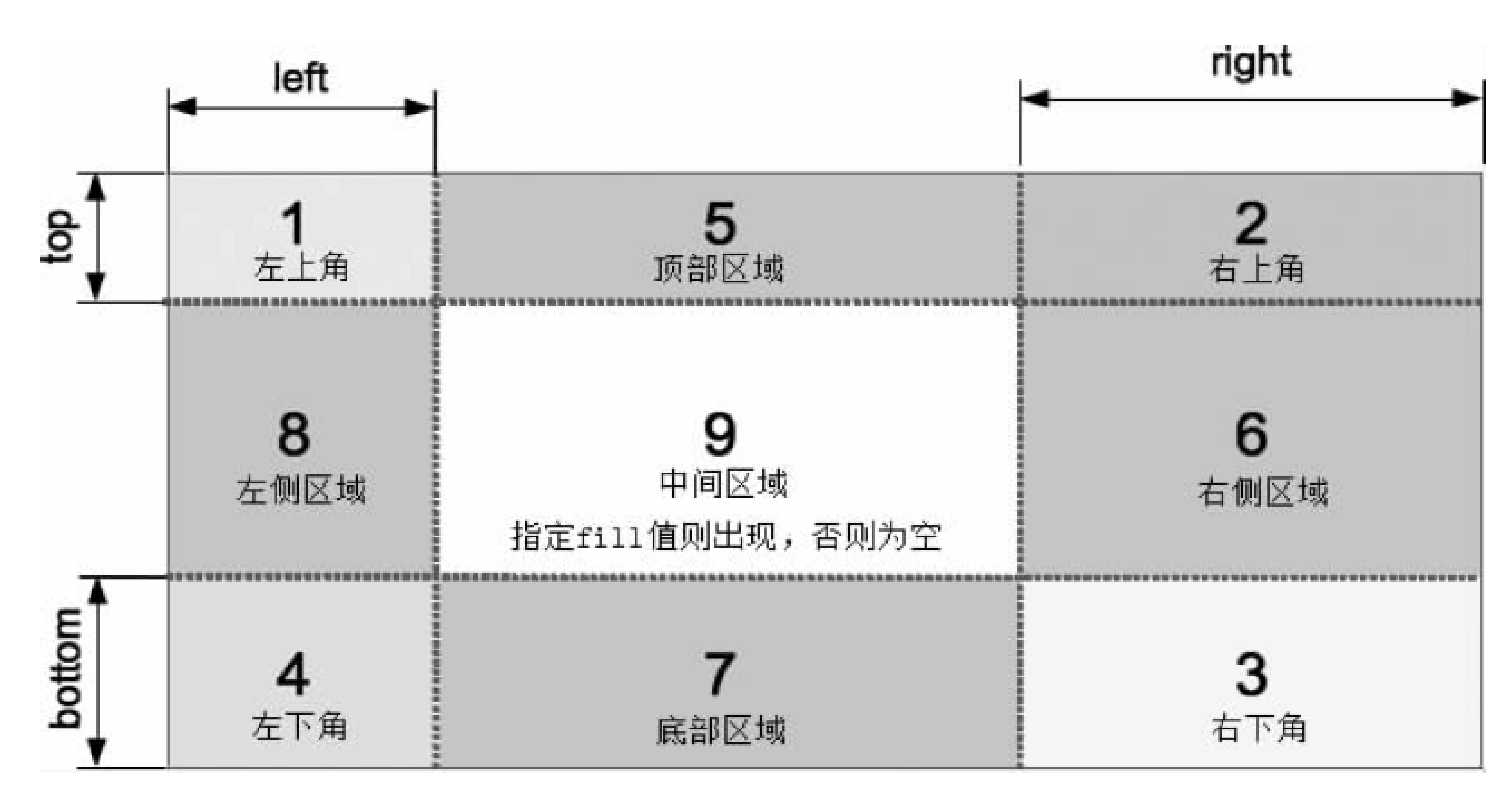


图 3-5　裁剪图片区域示意图

再举个例子来说明一下 border-image-slice 属性值是如何用来切割图片的。如“border-image-slice：10 20 30 40”属性值分别指定了切割图片时上、右、下、左边框的像素，但这里不能使用单位。切割示意图如图 3-6 所示。

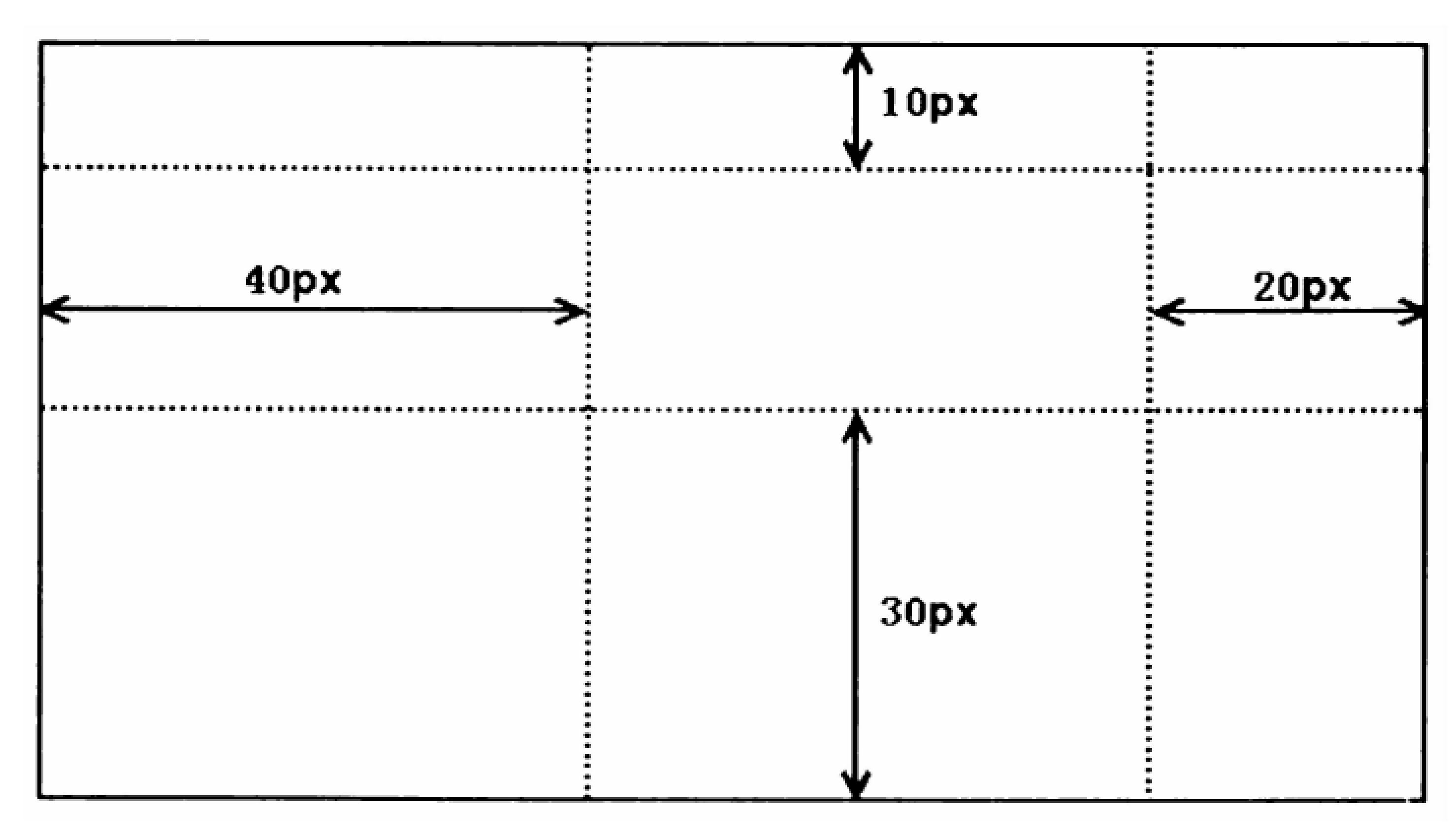


图 3-6　切割方向示意图

border-image-slice 属性值为 1 个值时，表示切割的图片上、下、左、右四个方向的像素值相同；当为 2 个值时，第 1 个值表示切割的图片上、下两个方向的像素值，第 2 个值表示切割的图片左、右两个方向的像素值；当为 3 个值时，第 1 个值表示切割的图片上方的像素值，第 2 个值表示切割的图片左、右两个方向的像素值，第 3 个值表示切割的图片下方的像素值；当为 4 个值时，则 4 个值依次按照上、右、下、左顺时针方向的切割图片像素值。这里再次强调，border-image-slice 属性的值不能添加任何单位。

【例 3-5】 定义一个图片边框。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>图片边框 border-image-slice 属性</title>
<style>
    .pic-border{
        width:300px;
        height:80px;
        border:30px solid #000;
        border-image-source:url(images/border.png);
        border-image-slice:27;
    }
</style>
</head>
<body>
    <div class="pic-border"></div>
</body>
</html>
```

程序的运行结果如图 3-7 所示。

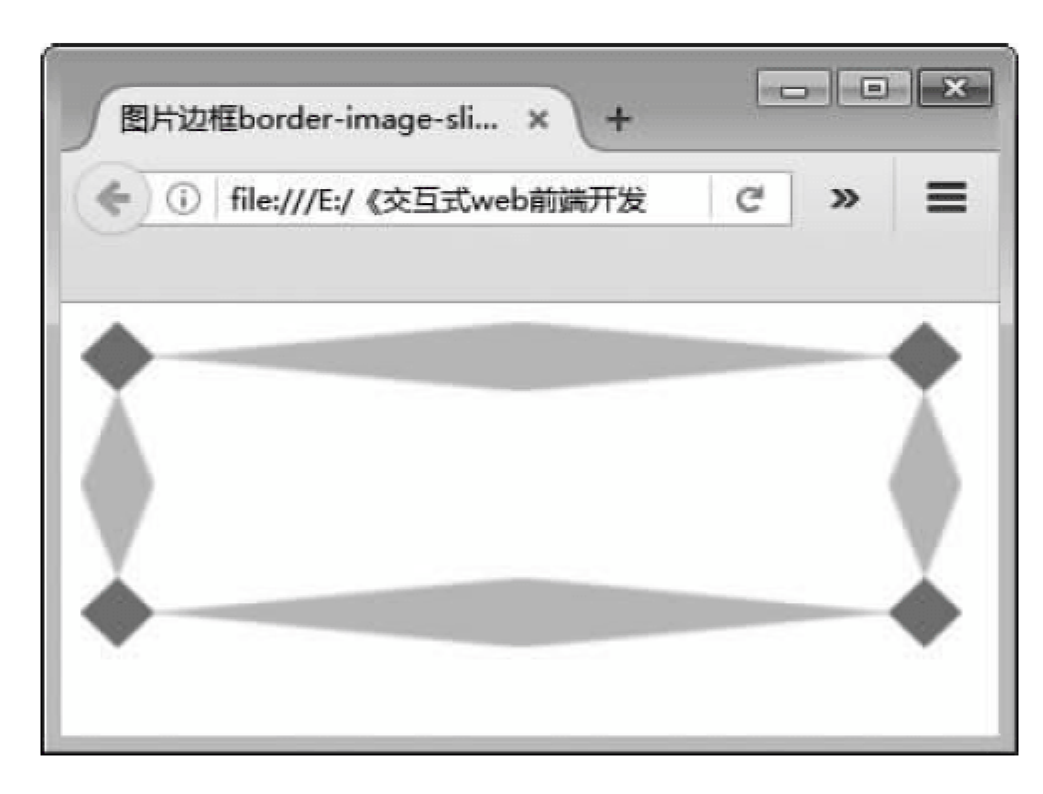


图 3-7　图片边框

border-image-slice 属性值不允许设置为负值，如果设置了负值或者设置值大于图片的高度或者宽度，都会用 100% 代替；border-image-slice 属性切割的区域有可能会重叠，如果右切和左切的值之和大于或等于图片的宽度，那么顶部区域和底部区域为空白，即图 3-5 中的 5 和 7 这两个区域会成为空白。

border-image-width 属性指定图片边框四个方向上的像素值。属性值可以指定 1～4 个像素值、百分比或数字。当属性值为数字时表示的是倍数，是设置的 border-width 的倍数。属性值还可以是 auto。

【例 3-6】 定义一个不同边框宽度值的图片边框。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>图片边框 border-image-width 属性</title>
<style>
    .pic-border{
        width:300px;
        height:80px;
        border:30px solid #000;
        border-image-source:url(images/border.png);
        border-image-slice:27;
        border-image-width:27px 2 10%27px;
    }
</style>
</head>
<body>
    <div class="pic-border"></div>
</body>
</html>
```

程序的运行结果如图 3-8 所示。



图 3-8　不同边框宽度值的图片边框

由此可见，例 3-6 中“border-image-width：27px 2 10％ 27px”将图片边框宽度的上(top)设置为 27 像素，右(right)设置为 border-width 的 2 倍即 60 像素，下(bottom)设置为整个区域高(height＋2×border＝140(px))的 10％即 14 像素，右(left)设置为 27 像素，因此这些值将代替 30px 成为图片边框的各个边的宽度值。

border-image-outset 属性用于设置图片边框向边框外延伸的距离。属性值可以指定 1～4 个像素值或数字，分别代表图片边框的上、右、下、左四个方向边框图片向外延伸的距离。

【例 3-7】 定义一个图片边框向外延伸的图片边框。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>图片边框 border-image-outset 属性</title>
<style>
    div{
        width:300px;
        height:80px;
        border:30px solid #000;
        margin-left:100px;
    }
    .pic{
        border-image-source:url(images/border.png);
        border-image-slice:27;
    }
    .pic2{
        border-image-source:url(images/border.png);
        border-image-slice:27;
        border-image-outset:2;
    }
</style>
</head>
<body>
    <div class="pic"></div>
    <div class="pic2"></div>
</body>
</html>
```

程序的运行结果如图 3-9 所示。

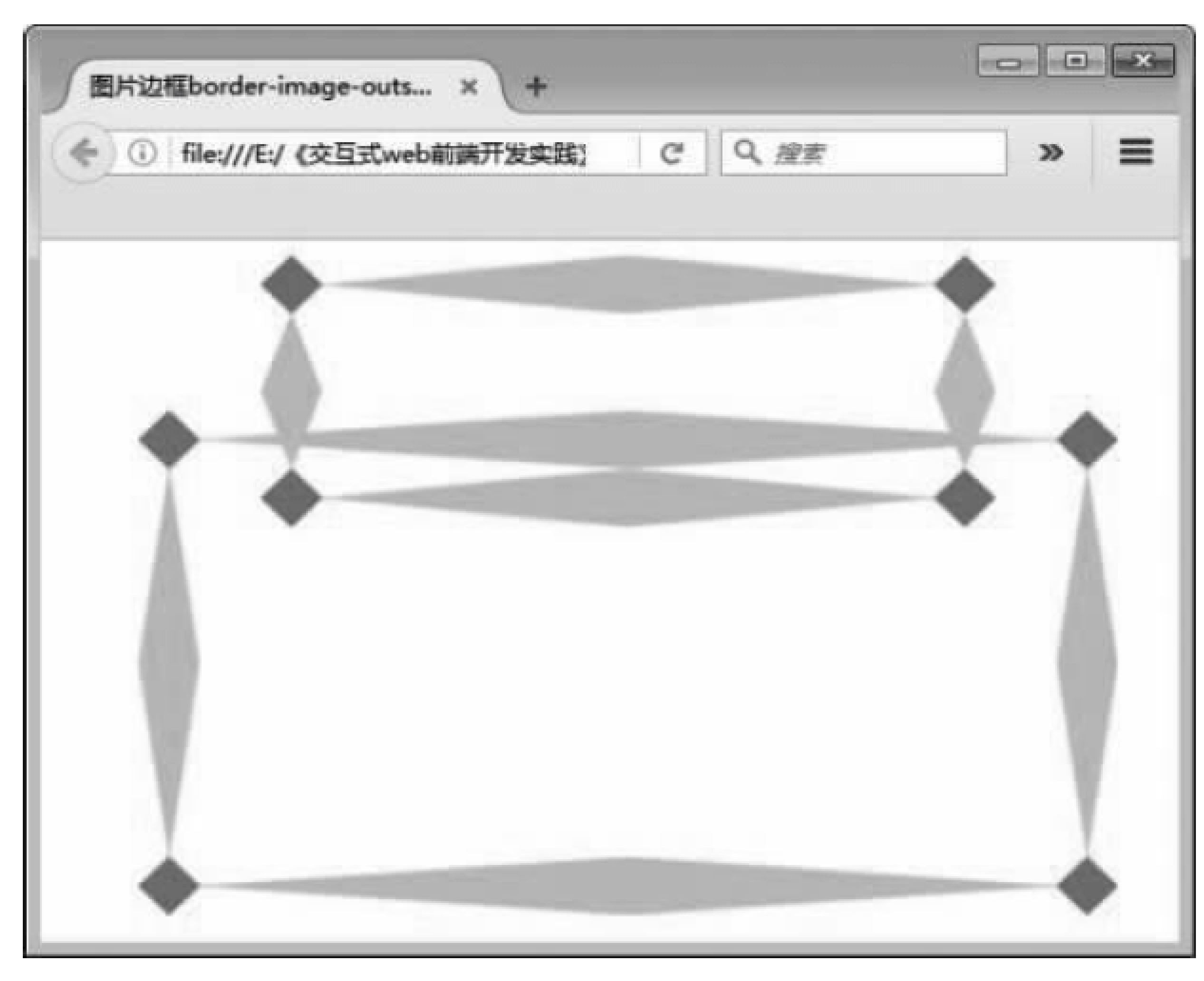


图 3-9　边框向外延伸的图片边框

例 3-7 中定义了两个大小相同的图片边框，第 2 个增加了 border-image-outset 属性，设置为 2，即图片边框向原边框外延伸 2 倍距离。因此，可以看到第 2 个图片边框的上边框向上延伸了 2 倍距离。原本的边框线应该紧挨着在第 1 个下边框的下面部分。border-image-outset 属性的展示效果如图 3-10 所示。



图 3-10　border-image-outset 属性示意图

border-image-repeat 属性用于控制边框图片的重复性。border-image-repeat 属性及描述见表 3-26。

表 3-26　border-image-repeat 属性及描述

属　性	描　　述
stretch	默认值，拉伸图片进行区域覆盖
repeat	重复平铺图片
round	取整平铺图片

border-image-repeat 属性的值可以同时指定 2 个值，分别表示横向、纵向的覆盖方式。上面三种属性值的显示示意图如图 3-11 所示。



图 3-11　border-image-repeat 属性示意图

当图片重复方式为 round 且平铺时，最后一个边框图片不能完全显示，且显示的区域不到图片的一半时，就不显示最后一张，然后扩大前面的图片，使它们完全覆盖显示区域；如果最后一张图片显示超过了一半，就稍微缩小前面的图片，让它们能完全显示并铺满边框区域。

(4) 外边框

外边框属性是网站开发人员为了向用户强调页面的某些方面，通过 outline 属性实现的外边框显示功能，通常用在表单控件中。外边框(outline)和物理边框(border)不同，它不占用页面布局中的空间，因此不影响页面的整体布局。外边框的样式像是总是浮于网页的上层，超脱出了网页这个平面，且它的形状不必是矩形的。外边框的属性及描述如表 3-27 所示。

表 3-27 外边框的属性及描述

属 性	描 述
outline-width	外边框线的宽度
outline-style	外边框线的样式
outline-color	外边框线的颜色
outline	复合属性,可以一次性定义外边框的宽度、样式和颜色

值得注意的是,外边框的所有边样式都是相同的,不能指定不同边为不同的样式。

outline-width 属性指定外边框线的粗细,属性值可以是长度值(数值＋单位)、thin、medium 或 thick,类似于 border-width 属性。

outline-style 属性指定外边框线的样式,属性值可以是 border-style 属性的属性值。

outline-color 属性指定外边框线的颜色,属性值是颜色值,可以是颜色名称、十六进制颜色或 RGB 值,类似于 border-color 属性。

outline 属性是一种复合属性,属性值可以一次性设置外边框的宽度、样式和颜色,中间用空格隔开,并且属性值之间可以采用任何顺序排列,不分先后顺序。

【例 3-8】 定义外边框。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>外边框</title>
<style>
    span{
        outline:solid #F63 5px;
    }
    input{
        outline-width:2px;
        outline-style:solid;
        outline-color:#6F3;
    }
</style>
</head>
<body>
    <p>请输入用户名,用户名必须是手机号码。</p>
    <p>请输入用户名,用户名必须是<span>手机号码</span>。</p>
    <input type="text">
    <button>提交</button>
</body>
</html>
```

程序的运行结果如图 3-12 所示。

通过例 3-8 的运行效果可以看到,"手机号码"添加的外边框并没有占位,它只是遮挡住了前后元素。因此,外边框不占据页面空间,不影响网页的布局。表单控件文本框的外边框通常用于获取焦点时,即当单击鼠标并进入文本框,从而可以输入文本时,则显示文本框的



图 3-12　外边框

外边框。

3. 伪类

CSS 3 引入了一些在选择器中添加的伪元素，我们称之为伪类。其实，CSS 2 中也可以使用伪类，只是支持程度有一定的限制。页面中常用到的伪类有“:focus”“:before”和“:after”。

(1) “:focus”伪类

在前面外边框中我们提到了获取焦点。“:focus”伪类就是元素获取焦点时的状态。在网页中链接和表单控件能够接收焦点。

通常情况下，当一个元素获取焦点时会让其显示的样式有所不同。如表单控件中的文本框，当文本框获取焦点时会显示其外边框；当失去焦点时，会隐藏外边框。将例 3-8 修改以后形成“:focus”伪类的用例。

【例 3-9】 定义“:focus”伪类。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>:focus 伪类</title>
<style>
    input:focus{
        outline:2px solid #6F3;
    }
</style>
</head>
<body>
    <p>请输入用户名,用户名必须是手机号码。</p>
    <input type="text">
    <button>提交</button>
</body>
</html>
```

程序的运行结果如图 3-13 和图 3-14 所示。



图 3-13　未获取焦点时



图 3-14　获取焦点时

由此可见，当鼠标光标移入文本框中能够在文本框中输入文字时，即文本框获取焦点时，显示定义的元素“:focus”伪类的样式。

（2）“:before”伪类、“:after”伪类和 content 属性

“:before”伪类是在定义的元素之前添加内容，“:after”伪类则是在定义的元素之后添加内容，并通过 content 属性添加。

【例 3-10】　在段落文字前面添加标题，在后面添加注释。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>:before 伪类、:after 伪类和 content 属性</title>
<style>
    p{
        font-family:"微软雅黑";
        font-size:16px;
        text-indent:32px;
    }
    p:before{
        content:"伪类定义";
        font-size:24px;
        font-weight:bold;
    }
    p:after{
        content:"(详细说明请参考本书中的介绍)";
        font-size:12px;
        color:#F00;
    }
</style>
</head>
<body>
    <p>":before"伪类是在定义的元素之前添加内容,":after"伪类则是在定义的元素之后添
    加内容,通过 content 属性添加。</p>
</body>
</html>
```

程序的运行结果如图 3-15 所示。



图 3-15 “:before”和“:after”伪类在元素之前和之后添加内容

由例 3-10 中可以看见“:before”伪类用于元素内容之前，“:after”伪类用于元素内容的末尾。都是通过样式属性 content 添加的内容。content 属性不仅仅可以添加文本内容，还可以添加其他的内容，如通过 url()添加图片或其他文件。

4. 字体阴影属性

CSS 3 为字体增加了阴影属性 text-shadow，可以设置字体的阴影显示效果。其语法格式如下：

```
text-shadow:xoffset yoffset radius color;
```

或者

```
text-shadow:color xoffset yoffset radius;
```

这两种语法格式都是正确的，指定的阴影颜色可以写在前面，也可以写在后面。text-shadow 属性及描述见表 3-28。

表 3-28 text-shadow 属性及描述

属　性	描　　述
color	指定阴影的颜色，如果省略，阴影颜色使用字体颜色作为引用颜色
xoffset	指定阴影的横向偏移
yoffset	指定阴影的纵向偏移
radius	指定阴影的模糊半径，半径越大则越模糊

以上四个属性值同时设置在属性 text-shadow 的属性值中，每个属性值之间用空格隔开。

【例 3-11】 定义文本的阴影显示效果。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
```

```
<title>字体阴影</title>
<style>
    h1{
        text-shadow:#F30 10px -10px 4px;
    }
</style>
</head>
<body>
    <h1>字体阴影</h1>
</body>
</html>
```

程序的运行结果如图 3-16 所示。



图 3-16　阴影文本

text-shadow 属性还可以设置多个阴影值,之间用逗号隔开。

【例 3-12】　定义一组阴影效果的文本显示。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>添加多个阴影</title>
<style>
    h1{
        text-shadow:10px 10px 2px #222,30px 30px 4px #555,50px 50px 6px #888;
    }
</style>
</head>
<body>
    <h1>字体阴影</h1>
</body>
</html>
```

程序的运行结果如图 3-17 所示。



图 3-17　具有多个阴影的文本显示

3.2.2　CSS 3 变形设置

CSS 3 新增了可以使元素的显示效果发生一定变化的属性 transform。在以往的样式设置中，对于元素的显示通常都是中规中矩的矩形显示样式，如果要设置一定的变形，通常都是用图片实现的。transform 属性可以设置元素的位移、缩放、旋转和倾斜后的显示样式。它的属性值采用的是变形函数，如表 3-29 所示。

表 3-29　transform 属性及描述

属　　性	描　　述
translate(tx[,ty])	设置横向、纵向位移，ty 省略时默认为 0
scale(sx[,sy])	横向、纵向上的缩放比。sy 可省略，如省略 sy，则默认等于 sx，即保持纵横比缩放
rotate(angle)	顺时针旋转 angle 角度
skew(sx[,sy])	沿 X 轴倾斜 sx 角度，沿 Y 轴倾斜 sy 角度，sy 省略则默认为 0

transform 属性有一个派生出来的属性即 transform-origin 属性，用于设置变形的中心点。它的属性值可以是 left、top、right、bottom、center、长度值或者百分比，分别表示变形中心点的坐标。如“transform-origin: left top;”表示元素区域左上角为变形的中心点。

1. 位移

translate(tx[,ty])变形函数设置元素区域的位移。参数 tx 表示横向位移，可以设置像素值或百分比，正值向右位移，负值向左位移；参数 ty 表示纵向位移，与参数 tx 类似，可以设置像素值或百分比，正值向下位移，负值向上位移，省略时，默认值为 0，即纵向上不发生位移。

2. 缩放

scale(sx[,sy])变形函数设置元素区域按横向或纵向上的缩放比例。参数 sx 表示横向上的缩放比例，设置某个具体浮点数表示缩放的倍数。参数 sy 可以省略，省略情况下 sy 的默认值等于 sx，即在保持纵横比例相同的情况下进行元素区域的缩放。

3. 旋转

rotate(angle)变形函数设置元素区域按照顺时针方向旋转 angle 角度。参数 angle 是

一个角度值，单位为度(deg)。一个圆形为 360deg。

4. 倾斜

skew(sx[,sy])变形函数设置元素区域沿 X 轴倾斜 sx 角度，沿 Y 轴倾斜 sy 角度，参数 sx 和 sy 都是一个角度值，单位为度(deg)。参数 sy 可以省略，省略情况下默认值为 0，即不发生 Y 轴的倾斜。

transform 属性可以同时指定多个变形函数，中间用空格隔开。例如：

```
transform:rotate(30deg) translate(260px, 60px) scale(2.4,0.4);
```

值得注意的是，变形函数的先后顺序会直接影响变形后的显示效果。同样的变形函数，使用的先后顺序不同，得到的效果也会有所不同。

【例 3-13】 transform 属性样式的变形设置。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>CSS3 变形设置</title>
<style>
    div{
        width:200px;
        height:100px;
        background:rgba(255,0,0,0.3);
        margin:30px;
    }
    .box{
         transform:rotate(30deg) translate(100px, 30px) scale(1.4,0.4) skew
         (30deg,120deg);
    }
    .box2{
         transform:translate(100px, 30px) scale(1.4,0.4) skew(30deg,120deg)
         rotate(30deg);
    }
</style>
</head>
<body>
    <div class="box"></div>
    <div class="box2"></div>
</body>
</html>
```

程序的运行结果如图 3-18 和图 3-19 所示。

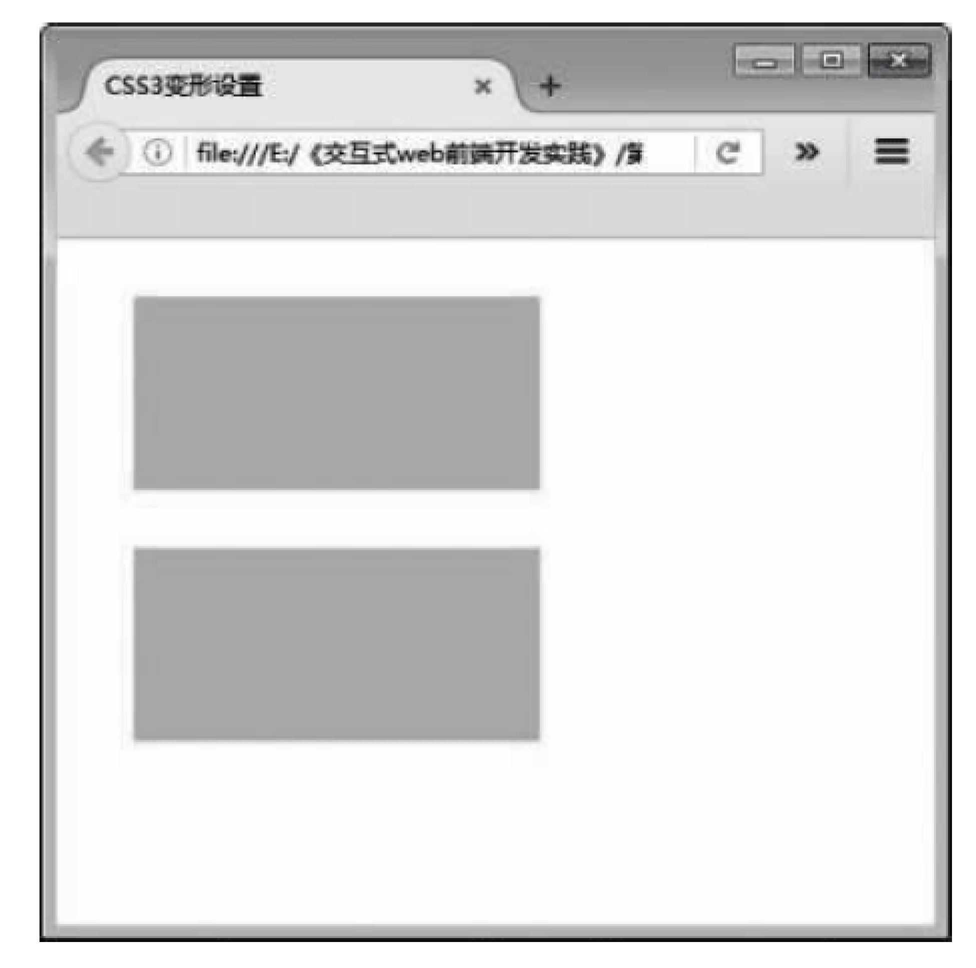



图 3-18　矩形未变形前

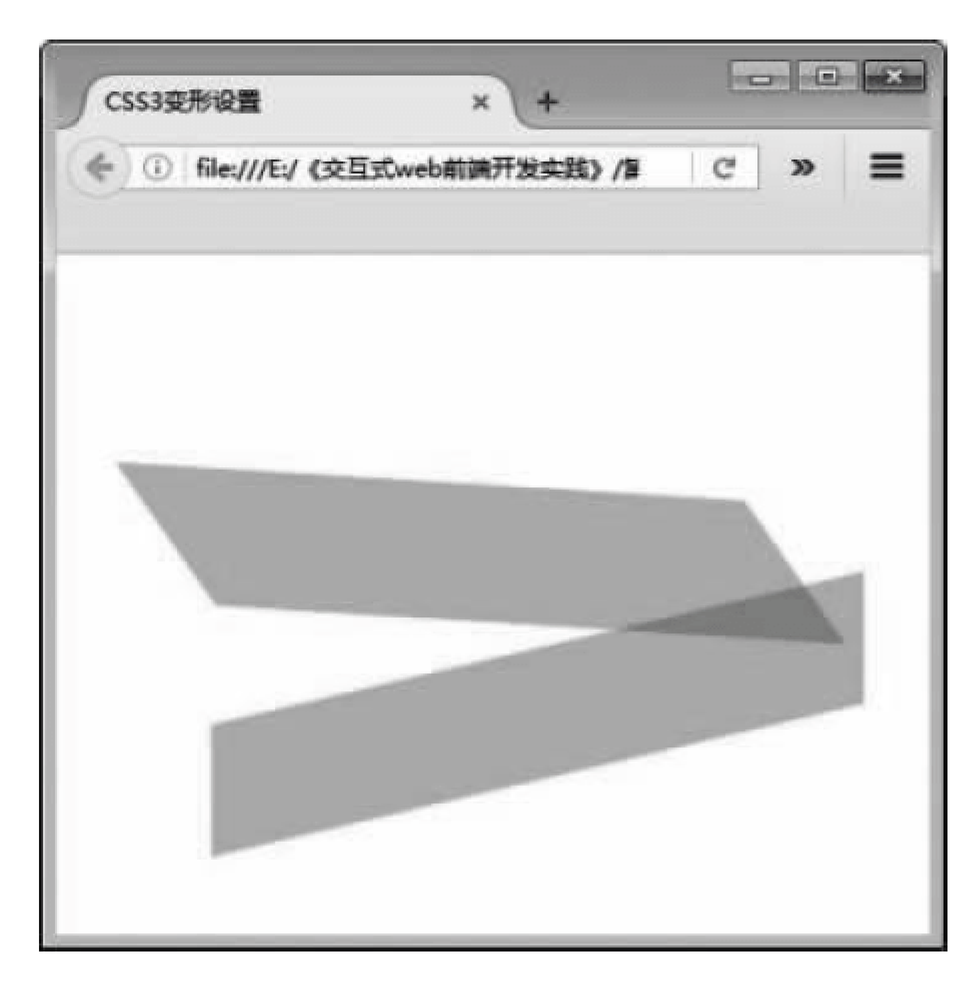


图 3-19　矩形变形后

这里定义了两个完全相同的矩形框，设置的变形函数也相同，只是前后顺序有所不同，得到的变形效果却是完全不同的显示样式。

提示： 在没有指定变形中心坐标的情况下，默认的变形中心都是元素区域的左上角。

3.2.3　CSS 3 动画设置

CSS 3 中新增了简单的动画设置，有 transition 动画和 animate 帧动画。下面分别进行介绍。

1. transition 动画

transition 动画通过 transition 属性实现，transition 属性及描述如表 3-30 所示。

表 3-30　transition 属性及描述

属　性	描　述
transition-property	指定元素的 CSS 属性进行平滑渐变处理。可以指定 background-color、width、height 等各种标准的 CSS 属性
transition-duration	指定属性平滑渐变的持续时间
transition-timing-function	指定渐变的速度。该部分的值有 ease(慢—快—慢)、linear(匀速)、ease-in(先慢后快)、ease-out(先快后慢)
transition-delay	指定延迟时间，该属性值可以省略

以上 4 个属性值可以一起设置在 transition 属性的值中，中间用空格隔开，用来定义 CSS 动画。还可以同时指定多组 transition 的属性值，之间用逗号隔开。

【例 3-14】 实现一个在页面上随鼠标光标漂移的气球。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
```

```
<title>transition 动画</title>
<style>
    img#target {
        position: absolute;
        /* 指定气球图片的 left、top 属性会采用平滑渐变的方式来改变 */
        transition: left 5s linear,top 5s linear;
    }
</style>
</head>
<body>
    <img id="target" src="images/balloon.gif" alt="气球"/>

    <script type="text/javascript">
        var target=document.getElementById("target");
        target.style.left="0px";
        target.style.top="0px";
        //为鼠标的按下事件绑定监听器
        document.onmousedown=function(evt)
        {
            //将鼠标事件的 X、Y 坐标赋给气球图片的 left、top
            target.style.left=evt.pageX+"px";
            target.style.top=evt.pageY+"px";
        }
    </script>
</body>
</html>
```

程序的运行结果如图 3-20 所示。



图 3-20　transition 动画

当在页面中按下鼠标左键，气球会随之移动过来。

2. animate 帧动画

animate 帧动画是通过 animation 属性和 keyframes 定义的，类似于用 flash 实现的帧动画。animate 属性及描述如表 3-31 所示。

表 3-31　animate 属性及描述

属　　性	描　　述
animation-name	指定动画名称
animation-duration	指定动画的持续时间
animation-timing-function	指定动画的变化速度
animation-delay	指定动画的延迟时间，可以省略
animation-iteration-count	指定动画的循环次数，可以省略

animation-name 属性用于定义一个关键帧，定义格式如下：

```
keyframes 关键帧名称{
    from|to|百分比{
        属性 1: 属性值 1;
        属性 2: 属性值 2;
        ⋮
    }
    ⋮
}
```

其中，from|to|百分比表示关键帧的位置。还可以设置同时改变多个属性的动画。

【例 3-15】　实现一个动画。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>animate 帧动画</title>
<style>
    @keyframes 'complex' {
    /* 定义动画开始处的关键帧 */
        0%{transform: rotate(0deg) scale(1);
           background-color: #f00;}
    /* 定义动画进行 40%时的关键帧 */
        40%{transform: rotate(720deg) scale(0.1);
           background-color: #ff0;}
    /* 定义动画进行 80%时的关键帧 */
        80%{transform: rotate(1080deg) scale(4);
           background-color: #f0f;}
    /* 定义动画进行 100%时的关键帧 */
        100%{transform: rotate(0deg) scale(1);
           background-color: #00f;}
    }
    @-webkit-keyframes 'complex' {
    /* 定义动画开始处的关键帧 */
        0%{-webkit-transform: rotate(0deg) scale(1);
                      background-color: #f00;}
    /* 定义动画进行 40%时的关键帧 */
        40%{-webkit-transform: rotate(720deg) scale(0.1);
```

```
                        background-color: #ff0;}
    /* 定义动画进行 80%时的关键帧 */
        80%{-webkit-transform: rotate(1080deg) scale(4);
                        background-color: #f0f;}
    /* 定义动画进行 100%时的关键帧 */
        100%{-webkit-transform: rotate(0deg) scale(1);
                        background-color: #00f;}
    }
    div{width:400px; height:40px; line-height:40px; background:#CCC;}
    div:hover {
            /* 指定执行 fkjava 动画 */
            animation-name: 'complex';
            -ms-animation-name: 'complex';
            -moz-animation-name: 'complex';
            -o-animation-name: 'complex';
            -webkit-animation-name: 'complex';
            /* 指定动画的执行时间 */
            animation-duration: 5s;
            -ms-animation-duration: 5s;
            -moz-animation-duration: 5s;
            -o-animation-duration: 5s;
            -webkit-animation-duration: 5s;
            /* 指定动画的循环次数为 1 */
            animation-iteration-count: 1;
            -ms-animation-iteration-count: 1;
            -moz-animation-iteration-count: 1;
            -o-animation-iteration-count: 1;
            -webkit-animation-iteration-count: 1;
    }
</style>
</head>
<body>
    <div>鼠标光标悬停则开始动画</div>
</body>
</html>
```

程序的运行结果如图 3-21 所示。



图 3-21　例 3-15 的执行结果

当鼠标光标移入 DIV 区域，则显示 animate 帧动画的显示效果。目前只有 chrome、opera、safari 浏览器支持 animate 属性，且需要在前面加-webkit-前缀。Firefox 浏览器尚不支持该属性。

3.3 布局基础

3.3.1 盒子模型

盒子模型是 CSS 中非常重要的核心内容，它是使用 CSS 控制页面元素的外观显示和位置定位的基础。网页文档中的每个元素都可以被视为一个盒子。可以理解为，网页布局实质上就是将大大小小的各种不同的盒子通过嵌套来进行合理的摆放。在布局的过程中，值得注意的是盒子实际占位的计算和是否兼容浏览器等问题的解决。一个标准的 W3C 盒子模型由 content（内容）、padding（填充，也称内边距）、margin（外边距）和 border（边框）这 4 个属性组成，如图 3-22 所示。

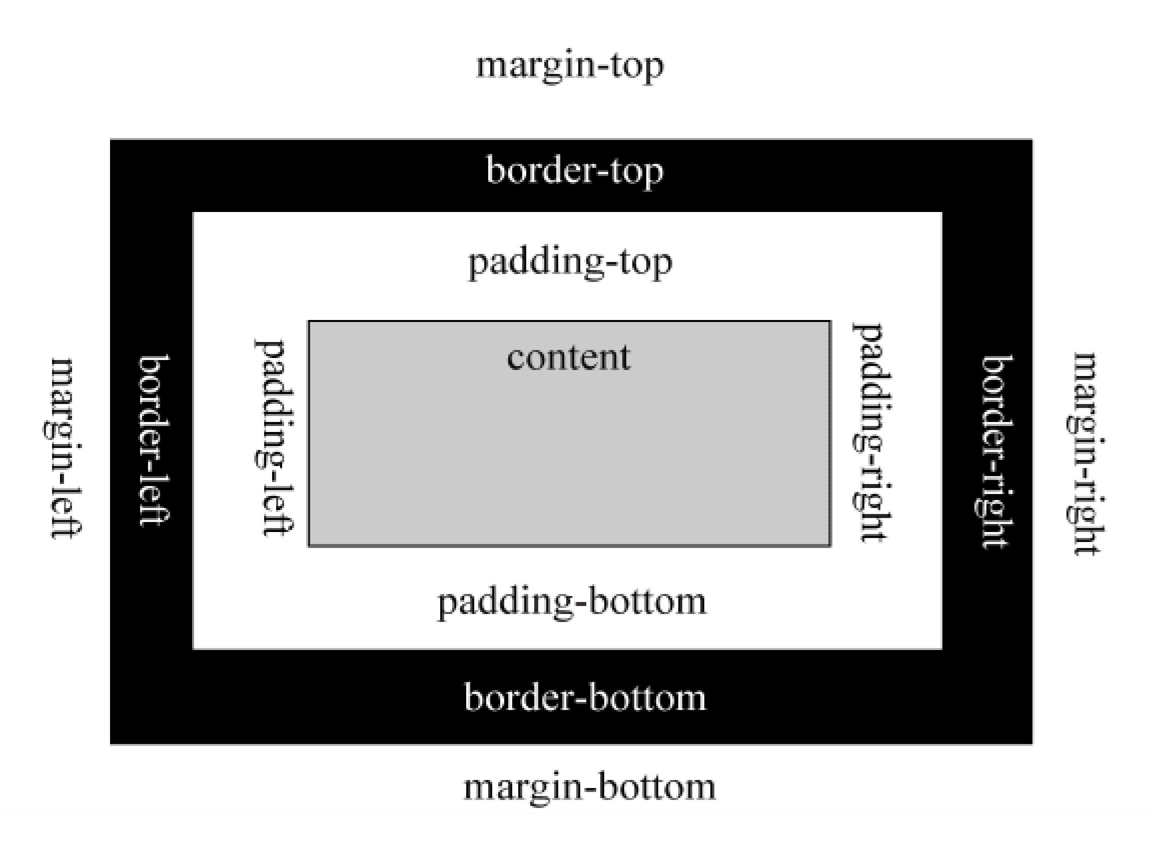


图 3-22　盒子模型

程序员可以通过生活中的盒子来理解样式设置中的盒子模型。content 就是盒子里装的东西，盒子一定会有宽度和高度；盒子外壳的厚度就是（border）边框；盒子里面的内容距盒子的边框会有一定的距离，这就是内边距（padding）；而盒子与盒子之间的距离就是外边距（margin）。下面分别进行介绍。

1. 宽度和高度

width 属性是指元素的内容在浏览器可视区域中的宽度，格式如下：

```
width:像素值/百分比;
```

属性值可以指定数值（如 100px）或者相对于父元素的百分比（如 80%）。这里的 width 设置的值与单个元素的宽度不同。元素的宽度包括元素的内容、内边距（填充）、边框和外边距。而 width 属性只为内容（content）实际的宽度。

height 属性指元素的内容在浏览器可视区域中的高度，与 width 属性类似，属性值可以指定数值（如 900px）或者相对于父元素的百分比（如 60%）。如果元素不指定高度，元素的高度一般为内容自身的高度，根据内容的多少而定，背景图片不一定会显示完全。

2. 边框

在现实生活中，一个盒子的边框具有一定的厚度、颜色和样式。网页中元素的边框（border）也包含同样的属性，具体有 border-width（边框的厚度）、border-style（边框的样式）和 border-color（边框的颜色）等属性，缺一不可。在 CSS 3 中，还可以为边框设置圆角（border-radius）以及盒子的阴影（box-shadow）等属性。border 属性前面介绍过，这里就不再赘述。

盒子的阴影是 CSS 3 新增的属性，在前面我们介绍过文本阴影的属性。盒子的阴影属性与文本阴影属性类似。语法格式为：

```
box-shadow:xoffset yoffset radius color inset/outset;
```

其中，参数 xoffset 为 X 轴的偏移位置，是一个像素值；参数 yoffset 为 Y 轴的偏移位置，是一个像素值；参数 radius 表示模糊度，也是一个像素值，数值越大越模糊；参数 color 是设置阴影的颜色；参数 inset/outset 是设置盒子阴影的位置（内阴影/外阴影），该参数可缺省，默认状态采用的是外阴影。应用如下：

```
box-shadow:5px 5px 7px #000 inset;
```

3. 内边距

内边距是内容（content）与边框（border）之间的距离，分为上、右、下、左 4 个方向上的距离。语法为：

```
padding:属性值;
```

属性值的取值可以是像素（px）、厘米（cm），也可以是百分比（%），但不允许为负值。取值个数可以是 1～4 个。取值为 1 个时表示 4 个方向的内边距都相同；2 个时分别表示上下和左右的距离；3 个时表示依次为上、左、右、下的距离；4 个时则表示按顺时针方向上、右、下、左各边的内边距。例如：

```
padding:5px;                 /* 上、下、左、右的内边距都是 5 像素 */
padding:5px 10px;            /* 上、下的内边距是 5 像素,左、右的内边距是 10 像素 */
padding:5px 10px 15px;       /* 上方的内边距是 5 像素,左、右的内边距是 10 像素,下方
                                的内边距是 15 像素 */
padding:5px 10px 15px 20px;  /* 上方的内边距为 5 像素,右方的内边距为 10 像素,下方的
                                内边距为 15 像素,左方的内边距为 20 像素 */
```

4. 外边距

外边距与内边距一样，同样分为上、右、下、左 4 个方向的距离，只是它是盒子与盒子之间的距离。margin-top 为元素距离顶边元素之间的距离；margin-right 为元素距离右边元素之间的距离；margin-bottom 为元素距离底边元素之间的距离；margin-left 为元素距离左

边元素的距离。语法与内边距一样，但不同的是，margin 的属性值可以设置为负值。margin 的属性值取值个数可以为 1～4 个，每种取值方式同 padding 一样，不再赘述。

盒子模型中，要理解一个元素的实际占位宽度和高度是如何计算的。元素实际在页面中所占的总宽度如下：总宽度＝内容设置的本身宽度（width）＋元素的边框厚度（border-width）＋元素的内边距（padding）＋元素的外边距（margin）。父元素所指定的宽度一定不能小于它里面所有子元素占位累加起来的宽度，否则排版会错位。同理，元素实际在页面中所占的总高度为：总高度＝内容设置或本身的宽度（height）＋元素的边框厚度（border-width）＋元素的内边距（padding）＋元素的外边距（margin）。对这个计算的理解，在布局中尤为重要。

3.3.2　布局方式

CSS 的定位与显示属性可以把一个 HTML 元素定位在网页中的任何位置。定位与布局密切相关。布局中常用的定位方式有标准流定位、相对定位、绝对定位和浮动定位，另加一个设置元素显示的层叠顺序的 z-index 属性。下面分别进行介绍。

1. 标准流定位

根据盒子模型，网页是由多个元素组成的，在没有为 HTML 元素添加任何与定位相关属性的前提下，浏览器会根据各个元素在 HTML 文档中出现的顺序，自上而下进行排列显示。描述得形象一点，可以把这样的方式称作“流”，即常说的标准流。也就是说标准流定位显示的位置由元素在 HTML 文件中的位置决定。

标准流定位是默认的网页布局方式。当删除其中的某个元素时，下面的元素会自动上移，填补删除的空间。块级元素、行内元素依据自己的显示属性按照在文档中的先后次序依次显示。如果是块级元素，就从上到下一个接一个地排列，行内元素就在行中水平布置，有嵌套关系也会显示出来。除非有特殊的指定，否则所有的元素都在普通流中定位。

2. 相对定位

相对定位（position：relative）是一个比较容易掌握的概念。相对定位是相对于自身在标准流中的位置（以元素左上角为初始点）进行垂直或水平位置上、下、左、右的偏移，然后元素就会改变自己的位置，而移动到重新定义的位置上。例如：

```
position:relative;
left:30px;
top:20px;
```

元素相对于自己的左上角向下偏移 20px，向右偏移 30px。需要注意的是，设置了相对定位的元素不仅偏移了某个距离，并且还占据着自己原本所占有的空间，可能会影响其他元素的显示。

如图 3-23 所示，对 Box 2 进行了相对定位，向右移动 20px，向下移动 20px，并且保留了自己原本的位子，但 Box 3 的位置并没有受到任何影响，只是内容被 Box 2 遮挡了。

3. 绝对定位

绝对定位与相对定位不同，对元素进行绝对定位，参照的元素位置是该元素的父级乃至祖先级元素，如果元素没有已定位的祖先元素，则会参照 body 元素的最左上角来定义。绝

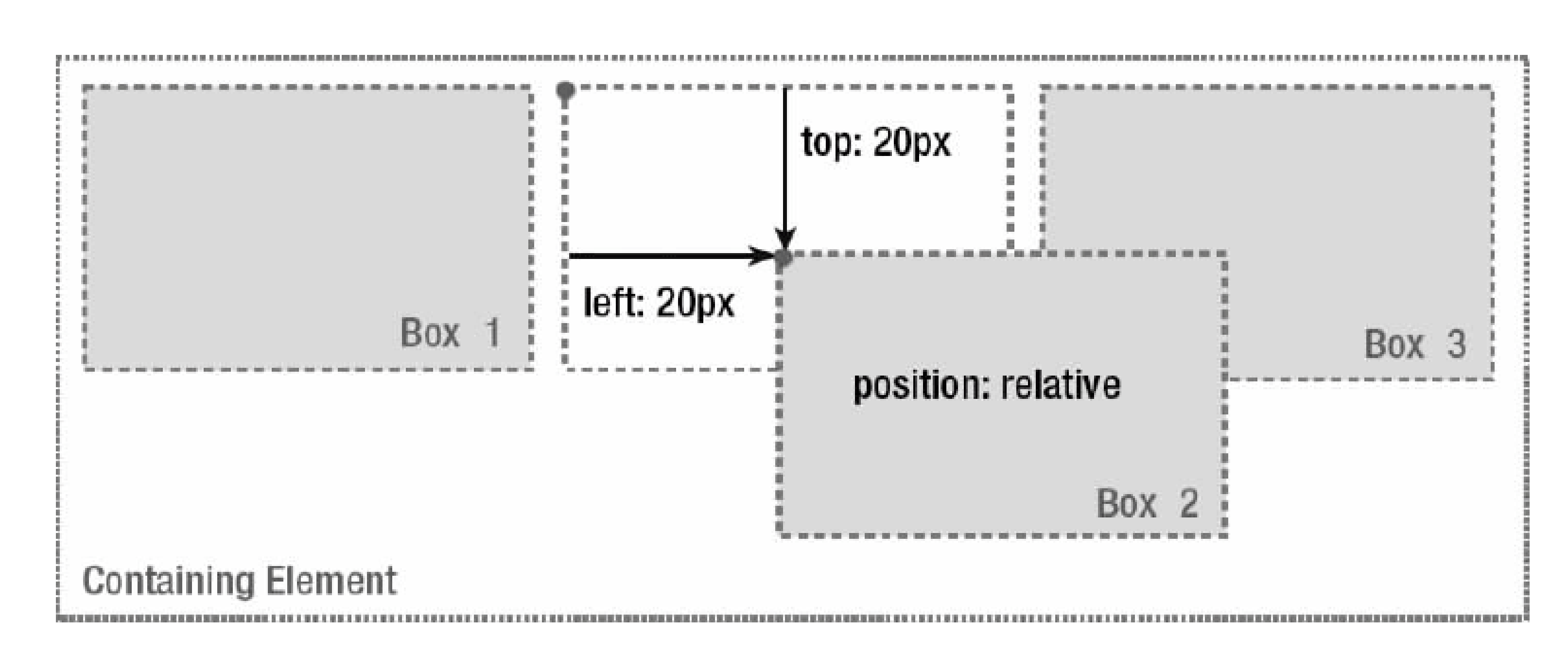


图 3-23　相对定位

对定位使元素的位置与标准流无关,因此不占据空间。绝对定位可以把某个元素精确地定位在页面的某个位置,使元素的位置与文档流无关,因此不占据标准流中的空间,对于普通文档流中的其他元素的布局,绝对定位的元素就像不存在一样。绝对定位的元素的位置是相对于最近的已定位的父元素而言的。如:

```
position:absolute;
left:30px;
top:20px;
```

图 3-24 所示为参照 Box 2 的父级元素(body)的左上角(必须是定义了定位属性的)向右偏移 20px,向下偏移 20px。但是原本在中间的 Box 2 会完全脱离标准流文档,并且影响了后面一个元素 Box 3 的显示位置。

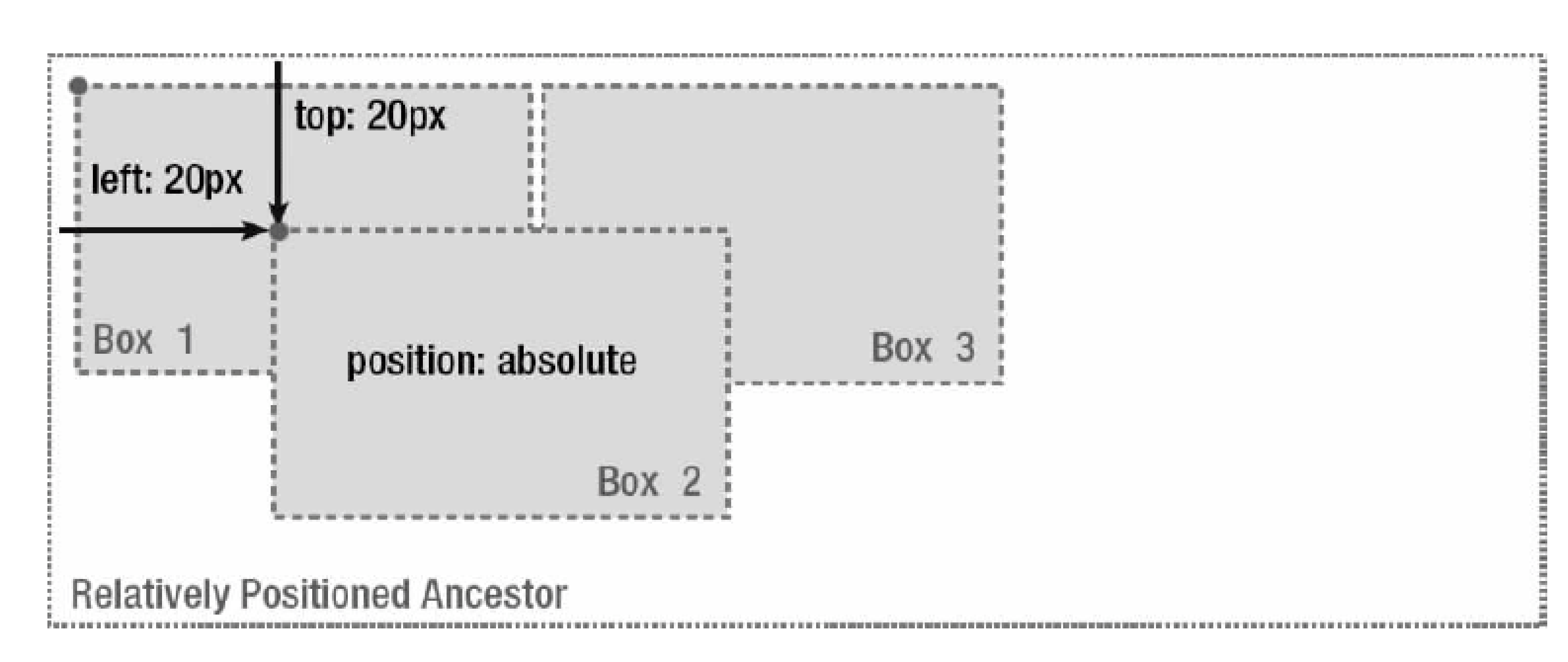


图 3-24　绝对定位

固定定位(position:fixed)是绝对定位的一种特殊情况,与绝对定位相似,唯一不同的是绝对定位是相对于已有定位属性的父元素区域的左上角,而固定定位则是固定在浏览器的视框位置。这样,当窗口滚动时,固定定位的元素不会随之滚动,总是出现在屏幕的固定位置。

4. 浮动定位

在页面的布局中经常会用到多栏目排版或图文混排的结构。float 属性可以实现布局中的多栏目并排及文字环绕的图文混排效果。

float 的属性及描述可以定义成如表 3-32 所示。

表 3-32　float 的属性及描述

属　性	描　　述
left	元素在包含框中向左浮动，包含框中的内容将移动到它的右边
right	元素在包含框中向右浮动，包含框中的内容将移动到它的左边
none	默认值。元素不浮动，会按照在标准流文档中的位置显示
inherit	从父元素继承 float 属性的值。IE 不支持该属性值

设置了 float 属性的元素是独立于标准流定位之外的可以左右移动，且尽可能远地放置在包含框（父元素）的边缘或另一个设置了浮动元素的边缘。换句话说，设置了浮动的元素，将不再处于标准流文档中，相当于浮在文档之上，不占据空间但会缩短行距（即浮动元素要占据文档中的宽度，但不占据高度），且可以左右移动，直到它的外边缘碰到包含框或另一浮动框的边缘进行定位，如图 3-25 所示。

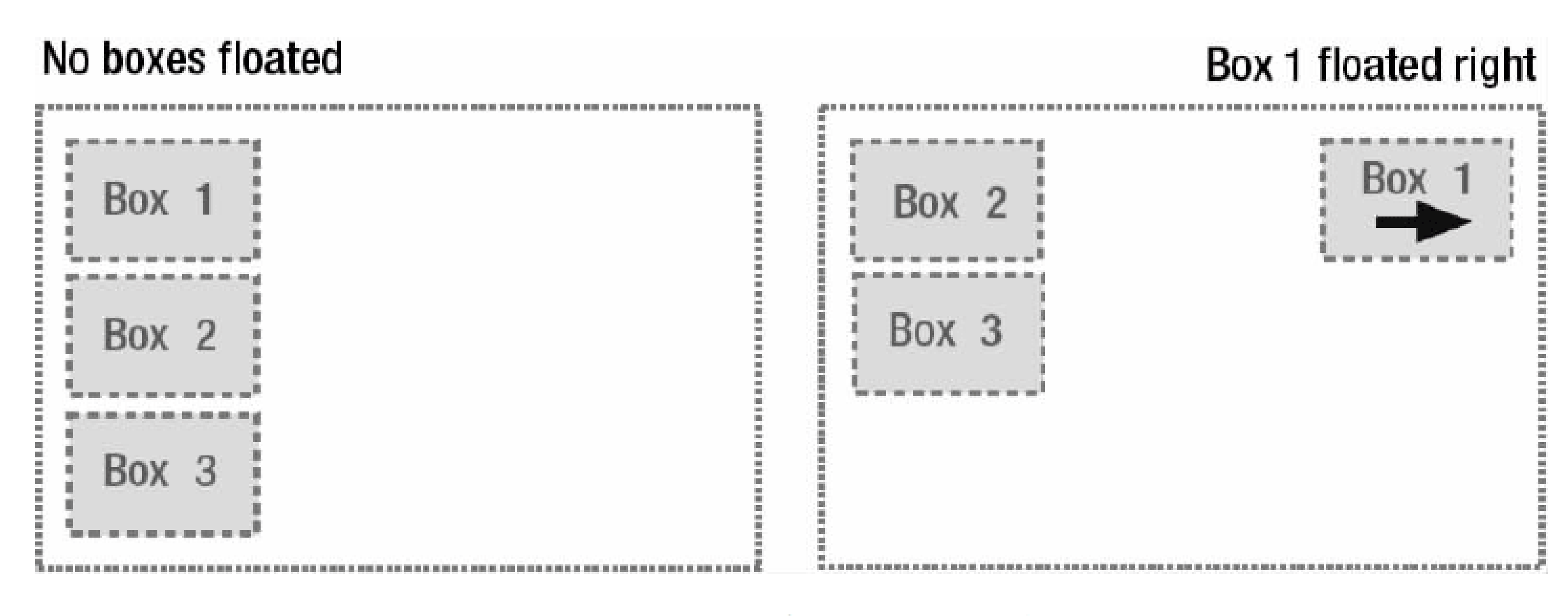


图 3-25　浮动定位的显示效果

任何 HTML 元素只要设置了 float 属性，就会产生浮动，并且元素的性质变成了行内块元素（即元素拥有了“display:inline-block;”的特性）。但对于标准流的元素来说，浮动元素不占高度，只有宽度。需要注意的是，浮动元素必须设置 width 属性，用于指定浮动框占用包含框的宽度，否则浮动框将占满包含框宽度的 100%，不会为浮动元素周围的其他元素余留空间，使包含框像是空白的块级元素，如图 3-26 所示。

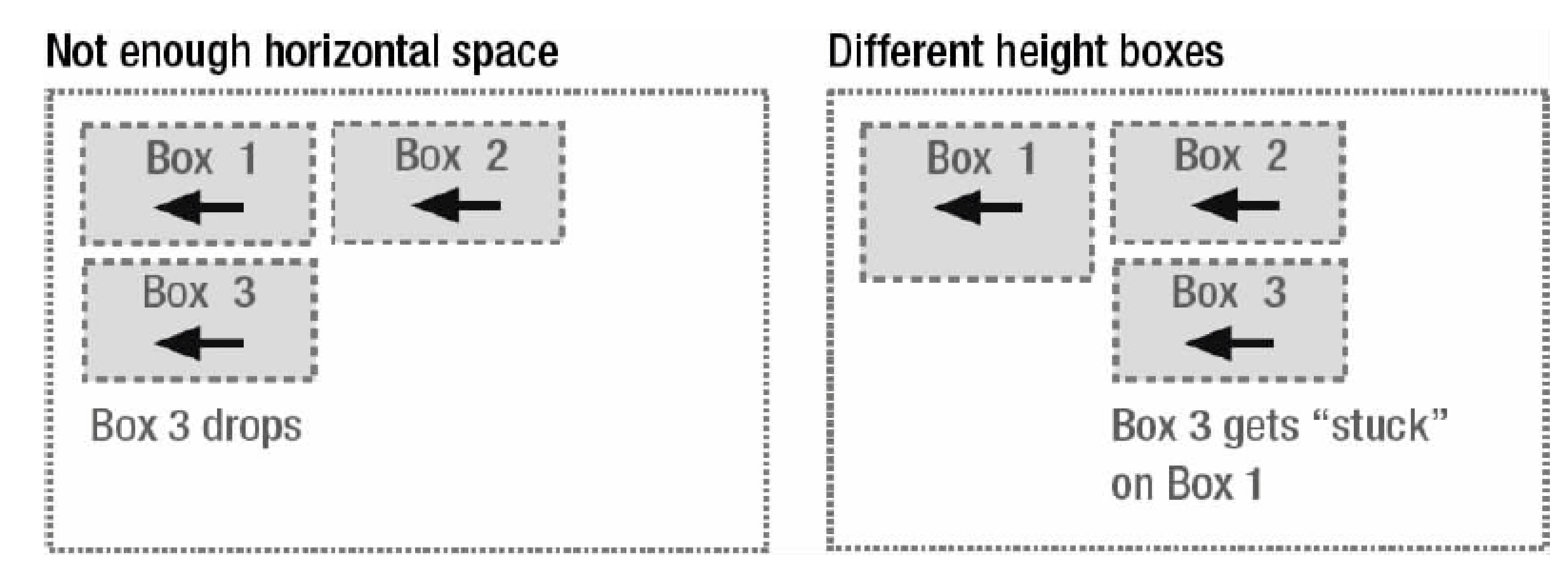


图 3-26　浮动元素包含框宽度不足及浮动元素高度不一致的显示效果

当使用浮动定位时，需要考虑包含框（父元素）的宽度是否能够水平并排其中的所有浮动元素。如果浮动元素的总宽度超出了父元素的宽度，超出的浮动元素会向下移动，另起一行按照规则进行排列。如果浮动元素的高度不同，当它们向下移动时，可能会移动到包含框的边缘，也有可能移动到另一浮动元素的边缘，如图 3-26 所示。

使用浮动定位要注意的两点。

(1) 当元素浮动时，它将不再处于普通文档流中，相当于浮在文档之上，不占据垂直空间，但是会缩短行框，产生文字环绕的效果。

(2) 当元素浮动时，它可以左右移动，直到它的外边缘碰到包含框或另一浮动框的边缘。

如果要为浮动元素留出垂直空间，使其他的元素不在其两侧显示，可以对其周围的元素使用清理属性 clear 属性。clear 的属性及描述如表 3-33 所示。

表 3-33　clear 的属性及描述

属性	描　述	属性	描　述
left	该元素的左侧不允许有内容	none	默认值。允许浮动元素出现在两侧
right	该元素的右侧不允许有内容	both	该元素的两侧均不允许有内容

设置了 clear 属性的元素，通过自动增加空白边，达到留出垂直空间的效果。

5. z-index 属性

z-index 属性用来设置元素的层叠顺序。当元素设置为相对定位或绝对定位时，经常会出现元素之间相互重叠的现象。这时，页面不再是一个平面，而具有了层次关系，有多个层面。也就是说页面向空间发展具有了 Z 轴，形成了一种三维结构。默认情况下，元素的层次关系为第一个元素位于后面元素的下方，即在结构上越后定义的元素越在上层。在页面布局中如果要打乱这种顺序，则需要 z-index 属性来实现。

z-index 属性只能用在具有相对定位或绝对定位的元素上才有效。它的属性值是一个数字，数值越大，元素越位于上层。

【例 3-16】 实现一个层叠顺序的显示效果。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>z-index 属性</title>
<style>
    div{
        width:200px;
        height:120px;
        position:absolute;
    }
    .box{
        background:#F00;
        left:150px;
        top:20px;
        z-index:9;
    }
    .box2{
        background:#FC3;
        left:100px;
```

```
        top:50px;
        z-index:5;
    }
    .box3{
        background:#F6F;
        left:50px;
        top:80px;
        z-index:1;
    }
</style>
</head>
<body>
    <div class="box">box1</div>
    <div class="box2">box2</div>
    <div class="box3">box3</div>
</body>
</html>
```

程序的运行结果如图 3-27 所示。



图 3-27　用 z-index 属性设置层叠顺序

例 3-16 中如果按文档结构顺序，box1 应该在最底层；box3 在最上层，但是使用了 z-index属性将 box1 放在了最上层。

3.4　综合实例

3.4.1　需求分析

这里创建一个 Web 静态页面的综合案例，将对第 2 章和第 3 章的知识进行总结，并将该知识运用到 Web 页面中。下面以图 3-28 为例，完成 HTML 结构和 CSS 样式。

分析：页面制作都是按照从上往下、从左往右的方式去实现的。根据这种方式，上例中的页面可以分为头部、主体内容和版权三部分，各部分再从上往下、从左往右依次实现即可。



图 3-28　综合案例

3.4.2　实现源代码

实现案例中的页面效果，由于篇幅关系，下面只列出了部分的源代码。具体请参考案例资源中的例 3-17。

部分 HTML 代码。

```
<div class="nav">
  <div class="logo"></div>
    <ul>
      <li class="on">首页</li>
        <li>公司介绍</li>
          <li>供应产品
            <ul class="subnav">
                <li><a href="#">压型板</a></li>
                  <li><a href="#">楼承板</a></li>
```

```
                    <li><a href="#">檩条</a></li>
                </ul>
            </li>
            <li>公司动态</li>
            <li>行业动态</li>
            <li>我们的优势</li>
            <li>公司资质</li>
            <li>联系我们</li>
        </ul>
    </div>
    ⋮
        <div class="center">
            <div class="header">
                <ul>
                    <li id="one">公司介绍 news</li>
                    <li id="two"><a href="#">more>></a></li>
                </ul>
            </div>
            <div class="jieshao">
                <ul>
                    <li><a href="#">资本挥师济南助力山东半岛地产 1 月份融资</a>
                    </li>
                    <li><a href="#">规模超预期货币政策将维持</a></li>
                    <li><a href="#">中国吸引资本流入底气不减抄底资产还是谋求转型?
                    </a></li>
                    <li><a href="#">房企纷纷进军老友相聚老歌传情资本邀投资人</a>
                    </li>
                    <li><a href="#">新春年会华丽开启资本 2014 盛世宏</a></li>
                    <li><a href="#">资本挥师济南助力山东半岛地产 1 月份</a></li>
                    <li><a href="#">规模超预期货币政策将维持</a></li>
                </ul>
            </div>
        </div>
    ⋮
    <div class="footer">
      <ul>
      <li><img src="images/fotter.png" width="554" height="52"></li>
          <li>  2014 重庆×××有限公司版权所有</li>
      </ul>
    </div>
    <div class="ask">
      <p>QQ 咨询</p>
      <p>免费电话</p>
      <p>售前咨询</p>
      <a>
          <font style="width:60px; height:1px; display:block"></font>
          <span>返回顶部</span>
      </a>
    </div>
```

部分 CSS 代码。

```
* {margin:0; padding:0;}
a{text-decoration:none;}
ul li{list-style-type:none;}
body{font-family:"微软雅黑"; overflow-x:hidden;}
a{ color:#000;}
input{color:#CCC;}
.main{width:100%; margin:0 auto; background:#FFF; height:auto;}
.nav{width:1127px; height:102px; background:#FFF; margin:0 auto; }
.nav .logo{ width:78px; height:72px; background:url(images/logol.png)
    no-repeat; float:left; margin-top:18px; margin-right:137px;}
.nav ul li{ float:left; margin:0 5px; height:65px; width:98px;
    line-height:65px; text-align:center; margin-top:37px;
    color:#01a1ca; font-family:"微软雅黑"; position:relative;}
.nav ul li ul.subnav{position:absolute; left:0; background:rgba(255,255,255,
    0.5);
    z-index:999; width:98px; height:120px; display:none;}
.nav ul li ul.subnav li{height:30px; margin:0; padding:0; color:#000;}
.nav.on{background:#01a1ca; color:#FFF;}
.footer{width:auto; height:100px; background:#01a1ca; margin:0 auto;}
.footer ul li{color:#FFF; font-size:12px; text-align:center; padding-
    top:10px;}
.ask{ width:80px; height:150px; position:absolute; right:0; top:0; z-index:
    999}
.ask p{width:80px; text-align:center; height:40px; line-height:40px;
    font-family:'微软雅黑'; font-size:14px; color:#676767; margin:0;
    padding:0; cursor:pointer; background:#fff;}
.ask a{display:inline-block; width:80px; height:60px; margin:10px 0 0 0;
    background:#fff; color:#000;  vertical-align:middle; cursor:pointer;}
.ask a span{ display:block; width:28px; height:44px; line-height:22px;
    font-family:'微软雅黑'; font-size:14px; line-height:22px;
    text-align:center; margin:8px 16px; _margin:-10px 0 0 16px;}
.ask a:hover{background:#01a1ca; color:#fff; zoom:1;}
.ask p:hover{background:#c40000; color:#fff;}
.ask p:cur{background:#c40000; color:#fff;}
```

本章小结

本章主要介绍了层叠样式表的应用。了解了 CSS 的编写规范、语法基础及如何运用到 HTML 页面中。学习了文本样式、背景样式、边框样式、列表样式及其他一些常用样式的设置。还介绍了 CSS 3 中新增的一些属性及变形样式和动画的设置。最后介绍了 Web 布局中最经典的盒子模型以及常用的几种定位方案。这些控制页面内容显示效果和位置定位的层叠样式表实现了结构与样式的完全分离。

第 4 章　JavaScript 编程基础

4.1　JavaScript 概述

JavaScript 是一种属于网络的脚本语言，是一种基于对象(Object)和事件驱动(Event Driven)并具有安全性能的脚本语言。已经被广泛用于 Web 应用开发，常用来为网页添加各式各样的动态功能，为用户提供更流畅美观的浏览效果。它是通过嵌入或调入在标准的 HTML 语言中实现的。JavaScript 与 HTML 超文本标记语言、Java 脚本语言(Java 小程序)一起实现在一个 Web 页面中链接多个对象，与 Web 客户交互作用，从而可以开发客户端的应用程序等。它的出现弥补了 HTML 语言的缺陷，它是 Java 与 HTML 折中的选择。

4.1.1　JavaScript 的特点

1. 一种解释性执行的脚本语言

JavaScript 也是一种解释性语言，它提供了一个非常方便的开发过程。JavaScript 的语法基本结构形式与 C、C++、Java 十分类似。但在使用前，不像这些语言需要先编译，而是在程序运行过程中被逐行地解释。JavaScript 与 HTML 标识结合在一起，从而方便用户的使用操作。

2. 一种基于对象的脚本语言

JavaScript 也可以被看作一种面向对象的语言，这意味着 JavaScript 能运用其已经创建的对象。因此，许多功能可以来自脚本环境中对象的方法与脚本的相互作用。

3. 一种简单弱类型的脚本语言

JavaScript 的简单性主要体现在：首先，JavaScript 是一种基于 Java 基本语句和控制流之上的简单而紧凑的设计，从而对于使用者学习 Java 或其他 C 语系的编程语言是一种非常好的过渡，而对于具有 C 语系编程功底的程序员来说，JavaScript 上手也非常容易；其次，其变量类型是采用弱类型，并未使用严格的数据类型。

4. 一种相对安全的脚本语言

JavaScript 作为一种安全性语言，不允许访问本地的硬盘，且不能将数据存入服务器，不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互，从而有效地防止数据的丢失或对系统的非法访问。

5. 一种事件驱动脚本语言

JavaScript 对用户的响应，是以事件驱动的方式进行的。在网页(Web Page)中执行了

某种操作所产生的动作，被称为“事件”(Event)。例如按下鼠标、移动窗口、选择菜单等都可以被视为事件。当事件发生后，可能会引起相应的事件响应，执行某些对应的脚本，这种机制被称为“事件驱动”。

6. 一种跨平台性脚本语言

JavaScript 依赖于浏览器本身，与操作环境无关，只要计算机能运行浏览器，并支持 JavaScript 的浏览器，就可正确执行，从而实现了“编写一次，走遍天下”的梦想。

因此，JavaScript 是一种新的描述语言，它可以被嵌入 HTML 文件中。JavaScript 语言可以做到响应使用者的需求事件(例如表单的输入)，而不需要任何的网络来回传输资料。所以当一位使用者输入一项资料时，此资料数据不用经过传给服务器(Server)处理再传回来的过程，而直接可以被客户端(Client)的应用程序所处理。

4.1.2　JavaScript 的优点及缺点

1. JavaScript 的优点

(1) JavaScript 减少网络传输。

在 JavaScript 这样的客户端脚本语言出现之前，传统的数据提交和验证工作均由客户端浏览器通过网络传输到服务器上进行。如果数据量很大，这对于网络和服务器的资源来说实在是一种无形的浪费，而使用 JavaScript 就可以在客户端进行数据验证。

(2) JavaScript 方便操纵 HTML 对象。

JavaScript 可以方便地操纵各种页面中的对象，用户可以使用 JavaScript 来控制页面中各个元素的外观、状态甚至运行方式，JavaScript 可以根据用户的需要“定制”浏览器，从而使网页更加友好。

(3) JavaScript 支持分布式运算。

JavaScript 可以使多种任务仅在客户端就可以完成，而不需要网络和服务器的参与，从而支持分布式的运算和处理。

2. JavaScript 的局限性

(1) 各浏览器厂商对 JavaScript 的支持程度不同。

目前在互联网上有很多浏览器，如 Firefox、Internet Explorer、Opera 等，但每种浏览器支持 JavaScript 的程度是不一样的，不同的浏览器在浏览一个带有 JavaScript 脚本的主页时，由于对 JavaScript 的支持稍有不同，其效果会有一定的差距，有时甚至会显示不出来。

(2) “Web 安全性”牺牲了 JavaScript 的一些功能。

当把 JavaScript 的一个设计目标设定为“Web 安全性”时，就需要牺牲 JavaScript 的一些功能。因此，纯粹的 JavaScript 将不能打开、读写和保存用户计算机上的文件。它有权访问的唯一信息就是该 JavaScript 所嵌入的那个 Web 主页中的信息，简而言之，JavaScript 只存在于它自己的小小世界——Web 主页里。

4.1.3　第一个 JavaScript 例子

【例 4-1】 创建第一个 JavaScript 程序。

在 HTML 网页上加入以下代码。

```
<html>
<head>
<meta charset="utf-8">
<title></title>
<script type="text/javascript">
    function displayDate(){
        document.getElementById("demo").innerHTML=Date();
    }
</script>
</head>
<body>
    <h1>我的第一个 JavaScript 程序</h1>
    <p id="demo">这是一个段落</p>
    <button type="button" onclick="displayDate()">显示日期</button>
</body>
</html>
```

执行以上代码，网页显示如图 4-1 所示。

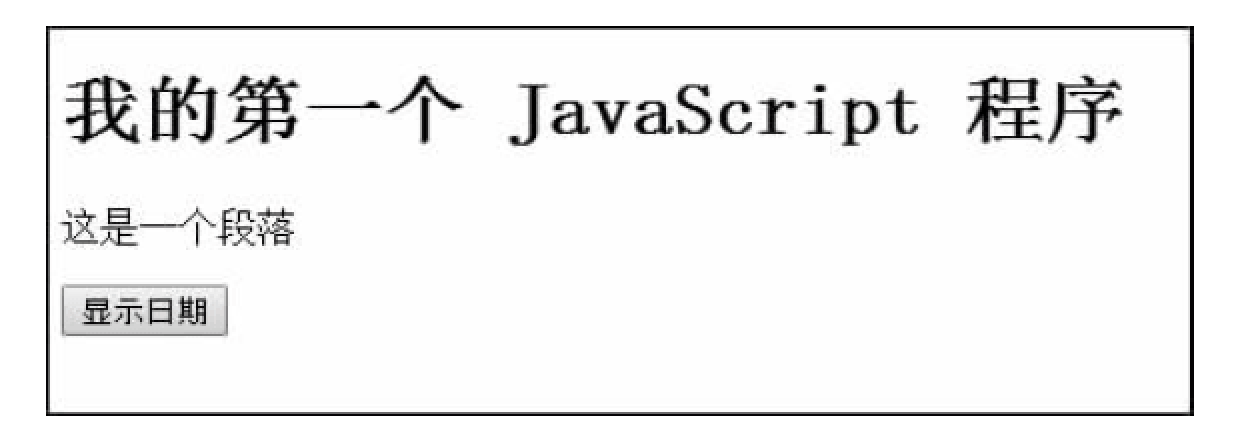


图 4-1　第一个 JavaScript 例子

HTML 中的脚本必须位于<script>与</script>标签之间。脚本可被放置在 HTML 页面的<body>或者<head>部分中。单击显示日期按钮，结果如图 4-2 所示。

我的第一个 JavaScript 程序

Tue Dec 06 2016 20:53:46 GMT+0800 (中国标准时间)

显示日期

图 4-2　创建第一个 JavaScript 程序显示日期的结果

4.2　JavaScript 语法

4.2.1　JavaScript 语句

JavaScript 语句是向浏览器发出的命令。语句的作用是告诉浏览器该做什么。JavaScript 用分号分割代码，通常在每条可执行的语句结尾添加分号。使用分号的另一个作用是在一行中编写多条语句。例如：

```
document.getElementById("demo").innerHTML="Hello World";
```

JavaScript 语句通过代码块的形式进行组合。代码块由左花括号开始,由右花括号结束。代码块的作用是使语句序列一起执行,JavaScript 函数是将语句组合在块中的典型例子。

【例 4-2】 代码块。

```
<script type="text/javascript">
    function myFunction(){
        document.getElementById("myPar").innerHTML="Hello World";
        document.getElementById("myDiv").innerHTML="How are you?";
    }
</script>
```

4.2.2 JavaScript 注释

JavaScript 注释可用于提高代码的可读性。

1. 单行注释

单行注释以//开头。

【例 4-3】 单行注释。

```
//输出标题
document.getElementById("myH1").innerHTML="Welcome to my Homepage";
//输出段落
document.getElementById("myP").innerHTML="This is my first paragraph.";
```

2. JavaScript 多行注释

多行注释以“/ * ”开始,以“ * /”结尾。

【例 4-4】 多行注释。

```
/*
  下面的这些代码会输出
  一个标题和一个段落
  并代表主页的开始
*/
document.getElementById("myH1").innerHTML="Welcome to my Homepage";
document.getElementById("myP").innerHTML="This is my first paragraph.";
```

4.2.3 变量与常量

JavaScript 命名规则:使用短名称(比如 x 和 y),也可以使用描述性更好的名称(比如 age、sum、totalvolume)。另外,遵循以下原则。

(1) 变量一般以字母开头。

(2) 变量也能以“ $ ”和“_”符号开头(不过不推荐这么做)。

(3) 变量名称对大小写敏感(y 和 Y 是不同的变量),JavaScript 语句和 JavaScript 变量

都对大小写敏感。

1. 变量

我们使用 var 关键词来声明变量。例如：

```
var pi=3.14;
var name="Bill Gates";
```

2. 常量

我们使用 const 来定义，但是 IE 浏览器不支持 const。例如：

```
const constantName=3.14;
```

4.2.4 运算符

JavaScript 运算符与其他语言的运算符类似。

1. 算术运算符

算术运算符主要用来进行算术运算，常用的算术运算符如表 4-1 所示。

表 4-1 常用的算术运算符

运算符	描述	实例(y=5)	结 果
+	加	x=y+2	x=7
-	减	x=x-2	x=3
*	乘	x=x*2	x=10
/	除	x=x/2	x=2.5
%	取余	x=x%2	x=1
++	自加	x=x++	x=6
--	自减	x=x--	x=4

2. 赋值运算符

赋值运算符主要用来赋值，常用的赋值运算符如表 4-2 所示。

表 4-2 常用的赋值运算符

运算符	描 述	实 例	结 果
=	直接赋值	x=3	x=3
+=	先加再赋值	x+=3	x=x+3
-=	先减再赋值	x-=3	x=x-3
*=	先乘再赋值	x*=3	x=x*3
/=	先除再赋值	x/=3	x=x/3

3. 关系运算符

关系运算符主要用来进行比较判断，常用的关系运算符如表 4-3 所示。

表 4-3　常用的关系运算符

运算符	描　　述	实　例	结果
==	等于	2==3	false
===	恒等于(值和类型都要比较)	2===3	false
!=	不等于	2!=3	true
>	大于	2>3	false
<	小于	2<3	true
>=	大于等于	2>=3	false
<=	小于等于	2<=3	true

4. 逻辑运算符

逻辑运算符主要用来进行逻辑判断，常用的逻辑运算符如表 4-4 所示。

表 4-4　常用的逻辑运算符

运算符	描　　述	实例(x=5)	结果
&&	逻辑与(and)	x>2&&x<9	true
||	逻辑或(or)	x>2||x<0	true
!	逻辑非，取逻辑的反面	!(x>2)	false

5. 三元运算符

格式：

```
表达式 1? 表达式 2: 表达式 3
```

说明：如果“表达式 1”的结果为真，执行“表达式 2”，否则执行“表达式 3”。

例如：

```
var a=2;
var b=3;
var x=a>b?a:b;
```

因为 2>3 条件为假，则执行“表达式 3”，即将 3 赋值给 x，所以 x 的结果为 3。

6. 字符串连接运算符

字符串连接运算符主要用于连接两个字符串或字符串变量。因此，在对字符串或字符串变量使用该运算符时，并不是对它们做加法计算。

例如：

```
var x="hello";
var y="world";
var s=x+y;          //s 的值为"helloworld"
```

4.2.5　正则表达式

1. 正则表达式的作用

(1) 测试字符串的某个模式。例如，可以对一个输入字符串进行测试，以确定该字符串

中是否存在一个电话号码模式或一个信用卡号码模式。这称为数据有效性验证。

(2) 替换文本。可以在文档中使用一个正则表达式来标识特定的文字,然后可以全部将其删除,或者替换为别的文字。

(3) 提取一个子字符串。可以用来在文本或输入字段中查找特定的文字。

2. 正则表达式的语法

一个正则表达式就是由普通字符(例如字符 a 到 z)以及特殊字符(称为元字符)组成的文字模式。该模式描述在查找文字主体时待匹配的一个或多个字符串。正则表达式作为一个模板,将某个字符模式与所搜索的字符串进行匹配。

(1) 两个特殊的符号"^"和"$"。它们的作用是分别指出一个字符串的开始和结束。

"^i":以 i 开始的字符串,例如"iPhone"等。

"script$":以$结束的字符串,例如"javascript$"等。

"^abc$":以 abc 开始并且以 abc 结束的字符串。

"*"、"+"和"?":表示一个或一系列字符重复出现的次数。它们分别表示“没有或更多”“一次或更多”还有“没有或一次”。例如:

- "ab*":表示一个字符串有一个 a 且后面跟着零个或若干个 b。比如:"a"、"ab"、"abbb"等。
- "ab+":表示一个字符串有一个 a 且后面跟着至少一个 b 或者更多。
- "ab?":表示一个字符串有一个 a 且后面跟着零个或者一个 b。
- "a?b+$":表示在字符串的末尾有零个或一个 a 且跟着一个或几个 b。比如:"6666abb"、"222b"等。

(2) {}用于表示重复次数的范围。

"ab{2}":表示一个字符串有一个 a 且跟着 2 个 b。比如:"abb"。

"ab{2,}":表示一个字符串有一个 a 且跟着至少 2 个 b。

"ab{3,5}":表示一个字符串有一个 a 且跟着 3～5 个 b。

(3) "¦"表示“或”操作。

"hi¦hello":表示一个字符串里有"hi"或者"hello"。

"(b¦cd)ef":表示"bef"或"cdef"。

"(a¦b)*c":表示一串若干 a 或 b 混合的字符串后面跟一个 c。比如:"aaac"。

(4) "."表示任意字符。

"a.[0-9]":表示一个字符串有一个 a 后面跟着一个任意字符和一个数字。比如:"ab2"、"a12"等。

"^.{3}$":表示有任意三个字符的字符串(长度为 3 个字符)。

(5) "[]"表示某些字符允许在一个字符串中的某一特定位置出现。

"[ab]":表示一个字符串有一个 a 或 b(相当于"a¦b")。

"[a-d]":表示一个字符串包含小写的 a 到 d 中的一个字母(相当于"a¦b¦c¦d"或者"[abcd]")。

"^[a-zA-Z]":表示一个以字母开头的字符串。

"[0-9]%":表示一个百分号前有一位的数字。

",[a-zA-Z0-9]$":表示一个字符串以一个逗号后面跟着一个字母或数字结束。

在方括号里用"^"表示不希望出现的字符,"^"应在方括号里的第一位。如:"%[^a-zA-Z]%"表示两个百分号中不出现字母。

(6) 正则表达式预定义类。

正则表达式预定义类已定义好固定的格式,可直接使用,常用的预定义类如表 4-5 所示。

表 4-5　常用的预定义类

类	等　价　于	说　　明
\d	[0-9]	匹配数字
\D	[^0-9]	匹配非数字
\s	[\n\r\t\f\x0B]	匹配一个空白字符
\S	[^\n\r\t\f\x0B]	匹配一个非空白字符
\w	[a-zA-Z0-9_]	匹配字母、数字和下划线
\W	[^a-zA-Z0-9_]	匹配除字母、数字、下划线之外的字符

3. 常用的正则表达式

(1) 只能输入数字,例如"^[0-9]* $"。

(2) 只能输入 n 位的数字,例如"^d{n} $"。

(3) 只能输入至少 n 位的数字,例如"^d{n,} $"。

(4) 只能输入 m~n 位的数字,例如"^d{m,n} $"。

(5) 只能输入长度为 3 的字符,例如"^.{3} $"。

(6) 只能输入由 26 个英文字母组成的字符串,例如"^[A-Za-z]+ $"。

(7) 只能输入由数字、26 个英文字母或者下划线组成的字符串,例如"^w+ $"。

(8) 验证用户密码,例如"^[a-zA-Z]w{5,17} $"正确格式为:以字母开头,长度为 6~18 位,只能包含字符、数字和下划线。

(9) 验证是否含有以下字符:^ % & ' , ; = ? $ 。

(10) 验证 E-mail 地址,例如"^"w+([—+.]"w+) * @"w+([—.]"w+) * "."w+([—.]"w+) * $ 。

(11) 验证一年的 12 个月,例如"^(0? [1-9]|1[0-2]) $"正确格式为:"01"~"09"和"1"~"12"。

【例 4-5】 以字母开头,长度为 6~18 位,只能包含字符、数字和下划线。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script language="JavaScript">
function isPass()
{
    var txt=document.getElementById("txt").value;
    var patrn=/^[a-zA-Z]\w{5,17}$/;
    if(patrn.exec(txt)!=null)
```

```
    {
        document.getElementById("p1").innerHTML="密码正确";
    }
    else
    {
        document.getElementById("p1").innerHTML="密码错误";
    }
}
</script>
</head>
<body>
<form>
    请输入密码:<input type="text" id="txt"/>
    <input type="button" onClick="isPass()" value="验证用户名"/>
</form>
    <p id="p1" style=" font-weight:bolder"></p>
</body>
</html>
```

输入错误格式的密码,结果如图 4-3 所示。

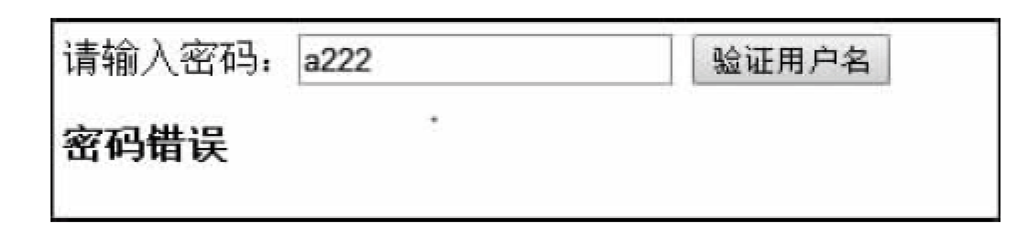


图 4-3　错误格式的密码

输入正确格式的密码,结果如图 4-4 所示。

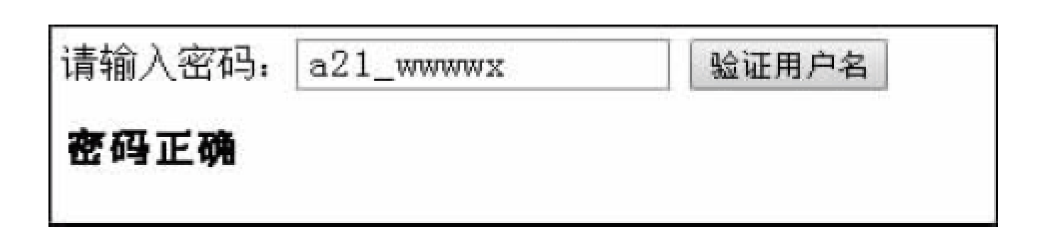


图 4-4　正确格式的密码

4.3　JavaScript 函数

4.3.1　函数的定义

JavaScript 的函数有两种写法:定义式和变量式。使用 function 来定义函数。

1. 定义式

格式:

```
function 函数名(参数 1,参数 2){
    //函数体;
}
```

定义式声明函数如下：

```
function myFun() {
    alert("我是 JavaScript 函数");
}
```

2. 变量式

JavaScript 函数可以通过一个表达式定义，函数表达式可以存储在变量中。

格式：

```
varfun=function(参数 1,参数 2){
    //函数体;
}
```

定义变量式声明函数如下：

```
varmyFun=function (){
    alert("我是 JavaScript 函数");
}
```

4.3.2 函数的参数及返回值

函数的参数可以没有也可以有多个，用逗号隔开，不用说明参数的类型。函数可以没有返回值，也可以有返回值，用 return 给出返回值。

【例 4-6】 带参数和返回值的函数。

```
functionmult(a,b) {
    rerurn a*b;
}
```

声明一个函数 mult，带有参数 a、b，将 a*b 的结果作为返回值。

4.3.3 函数的调用

1. 声明式函数的调用

通过函数名进行调用。

【例 4-7】 通过函数名调用。

```
<body>
<script type="text/javascript">
    functionclickTest(){
        alert("11");
    }
</script>
<input type="button" onclick=" clickTest ()" value="调用函数" />
</body>
```

单击“调用函数”按钮，调用 clickTest()函数，该函数弹出一个网页对话框，显示字符串

11，结果如图 4-5 所示。



图 4-5　调用函数的执行结果

2. 变量式函数的调用

变量式函数通过变量名调用，在函数表达式存储在变量后，变量作为一个函数被使用。

【例 4-8】　通过变量名调用函数。

```
<body>
<script type="text/javascript">
    var x=function (a,b){
        alert(a*b);
    }
</script>
<input name="" type="button" onclick="x(2,3)" value="调用函数" />
</body>
```

单击“调用函数”按钮，调用变量式函数 x(2,3)，该函数弹出一个网页对话框，对话框显示 2*3 的值。通过变量名调用的结果如图 4-6 所示，弹出的提示框中显示了 2*3 的值 6。

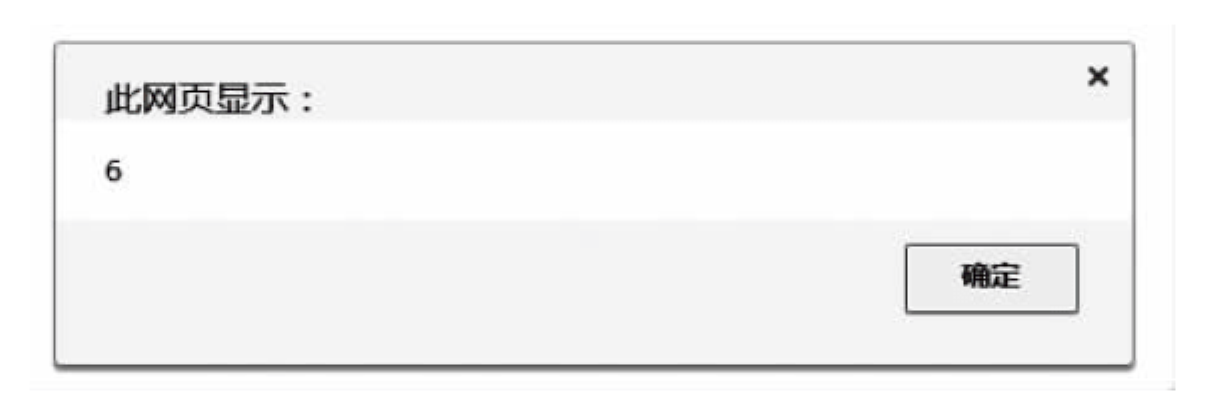


图 4-6　通过变量名调用

4.4　JavaScript 程序结构

4.4.1　顺序结构

程序顺序执行，依顺序逐条执行，只有在上一条语句执行完后，才能执行下一条语句。顺序结构示例如下：

```
<html>
<head>
<meta charset="utf-8">
```

```
<title></title>
<script type="text/javascript">
    var a=1;
    var b=2;
    var c=a*b;
    document.write("a*b="+c);
</script>
</head>
<body>
</body>
</html>
```

程序从第一条语句开始执行，直到最后一条语句。首先对变量 a 和 b 赋值，然后计算 a*b 的值并赋值给 c。然后在网页上显示“a*b=2”。

4.4.2　选择结构

通常在写代码时，总是为满足不同的需要来执行不同的动作。可以在代码中使用条件语句来完成该任务。

1. if 语句

只有当指定条件为 true 时，该语句才会执行相应的代码。

格式：

```
if (条件)
{
    //只有当条件为 true 时才执行的代码
}
```

【例 4-9】　当时间小于 20:00 时，下面的程序生成一个 Good day 的问候。

```
<script type="text/javascript">
    var time=15;
    if(time<20)
    {
        x="Good day";
    }
</script>
```

x 的结果是：

```
Good day
```

2. if-else 语句

if-else 语句在条件为 true 时执行第一部分代码，在条件为 false 时执行其他的代码。

格式：

```
if(条件)
{
    //当条件为 true 时执行的代码
```

```
}
else
{
    //当条件为 false 时执行的代码
}
```

【例 4-10】 当时间小于 20:00 时,将得到问候 Good day,否则将得到问候 Good evening。

```
<script type="text/javascript">
var time=21;
if(time<20){
    x="Good day";
}
else{
    x="Good evening";
}
</script>
```

执行程序后,x 的结果是:

```
Good evening
```

3. if-else if-else 语句

使用 if-else if-else 语句选择多个代码块之一来执行。

格式:

```
if(条件 1)
{
    //当条件 1 为 true 时执行的代码
}
else if(条件 2)
{
    //当条件 2 为 true 时执行的代码
}
else
{
    //当条件 1 和条件 2 为 false 时执行的代码
}
```

【例 4-11】 如果时间小于 10:00,则将发送问候 Good morning;否则如果时间小于 20:00,则发送问候 Good day,其他时间段发送问候 Good evening。

```
<script type="text/javascript">
    var time=21;
    if(time<10)
    {
        x="Good morning";
    }
```

```
    else if(time<20)
    {
        x="Good day";
    }
    else
    {
        x="Good evening";
    }
</script>
```

最终 x 的结果是：

```
Good evening
```

4．switch 语句

基于不同的条件来执行不同的动作。

格式：

```
switch(n) {
    case 1:
        //执行代码块 1 break;
    case 2:
        //执行代码块 2 break;
    default:
        //不满足 case 1 和 case 2 条件时执行的代码
}
```

【例 4-12】　显示今天是星期几。

```
<script type="text/javascript">
    var d=new Date().getDay();
    switch (d) {
        case 0:x="今天是星期日"; break;
        case 1:x="今天是星期一"; break;
        case 2:x="今天是星期二"; break;
        case 3:x="今天是星期三"; break;
        case 4:x="今天是星期四"; break;
        case 5:x="今天是星期五"; break;
        case 6:x="今天是星期六"; break;
    }
</script>
```

执行程序的结果是显示当日的星期，例如“今天是星期二”。

4.4.3　循环结构

循环结构是程序中一种很重要的结构。在给定条件成立时，按照一定的条件反复执行某程序段的算法步骤。给定的条件称为循环条件；反复执行的步骤称为循环体。

1. for 循环

格式：

```
for(语句 1; 语句 2; 语句 3)
{
    //被执行的代码块 4
}
```

语句 1 是初始化条件，语句 2 是循环条件，语句 3 是迭代部分，代码块 4 是循环体。

【例 4-13】 for 循环，打印 0～4 的数字。

```
<script type="text/javascript">
    for(var i=0; i<5; i++){
        var x="The number is "+i+"<br>";
        document.write(x);
    }
</script>
```

程序的执行结果如图 4-7 所示。

另外，语句 1 既可以有也可以没有。语句 1 可以定义多个变量。语句 2 也是可选的，省略语句 2 表示无限循环，通常需要与 break 一起使用。语句 3 也可以省略，例如可以将语句 3 放到循环部分执行。

【例 4-14】 省略语句 1 和语句 3 的 for 循环。

```
<script type="text/javascript">
    var i=1,len=5;
    for(; i<=len; ){
        document.write("The number is "+i+"<br>");
        i++;
    }
</script>
```

执行程序的结果如图 4-8 所示。

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4

图 4-7 打印 0～4 的数字

The number is 1
The number is 2
The number is 3
The number is 4
The number is 5

图 4-8 省略语句 1 和语句 3 的 for 循环

2. for/in 循环

JavaScript 中的 for/in 语句循环用于遍历对象的属性。

格式：

```
for(变量 in 对象)
{
    //循环逻辑代码的实现
}
```

【例 4-15】 for/in 循环。

```
<script type="text/javascript">
    var person={fname:"John",lname:"Doe",age:25};
    for(x in person){
        document.write(person[x]+" ");
    }
</script>
```

遍历对象，输出对象的属性值，程序执行的结果如图 4-9 所示。

```
John Doe 25
```

图 4-9　for/in 循环

3. while 循环

while 循环会在指定条件为真时循环执行代码块。

格式如下：

```
while(条件)
{
    //需要执行的代码
}
```

【例 4-16】 while 循环。

```
<script type="text/javascript">
    var i=1;
    while (i<5){
        x="The number is "+i+"<br>";
        document.write(x);
        i++;
    }
</script>
```

循环输出数字 1～4，结果如图 4-10 所示。

```
The number is 1
The number is 2
The number is 3
The number is 4
```

图 4-10　while 循环的执行结果

4. do-while 循环

do-while 循环是 while 循环的变体。先执行一遍循环体，然后判断条件，如果条件为真则继续执行循环体，如果条件为假则循环终止。

格式：

```
do
{
    //需要执行的代码
}
while(条件);
```

【例 4-17】 do-while 循环。

```
<script type="text/javascript">
    var i=1;
```

```
    do{
        x="The number is "+i+"<br>";
        document.write(x);
        i++;
    }
    while (i<5);
</script>
```

循环输出数字 1～4，结果如图 4-11 所示。

5. break 和 continue

break 和 continue 用于终止循环，break 语句用于跳出循环。break 语句跳出循环后，会继续执行该循环之后的代码(如果有的话)。

【例 4-18】 break。

```
<script type="text/javascript">
    var x="",i=0;
    for(i=0;i<10;i++){
        if(i==3){
            break;
        }
        x="The number is "+i+"<br>";
        document.write(x+"</br>");
    }
</script>
```

当 i=3 时结束循环，其他情况下不再执行，结果如图 4-12 所示。

```
The number is 1
The number is 2
The number is 3
The number is 4
```

图 4-11　do-while 循环的结果

```
The number is 0

The number is 1

The number is 2
```

图 4-12　运用 break 的结果

continue 语句中断循环中的迭代。如果出现了指定的条件，然后继续循环中的下一个迭代。

【例 4-19】 continue。

```
<script type="text/javascript">
    var x="",i=0;
    for(i=0;i<10;i++){
        if(i==3) {
            continue;
        }
        x="The number is "+i+"<br>";
        document.write(x+"</br>");
    }
</script>
```

当 i=3 时跳过了本次循环，其他情况继续输出，结果如图 4-13 所示。

```
The number is 0
The number is 1
The number is 2
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
```

图 4-13　运用 continue 的结果

4.5　异常处理

当 JavaScript 引擎执行 JavaScript 代码时，会发生各种错误，可能是语法错误，通常是程序员造成的编码错误或错别字；可能是拼写错误或语言中缺少的功能（可能由于浏览器差异）；可能是由于来自服务器或用户的错误输出而导致的错误；也可能是由于许多其他不可预知的因素。JavaScript 使用 try-catch 和 throw 来处理异常。

1．try-catch

try 语句允许定义在执行时进行错误测试的代码块。catch 语句允许定义当 try 代码块发生错误时所执行的代码块。JavaScript 语句 try 和 catch 是成对出现的。

格式：

```
try
{
    //在这里运行代码
}
catch(err)
{
    //在这里处理错误
}
```

【例 4-20】　用 try-catch 处理异常。

```
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script>
```

```
var txt="";
    function message(){
        try{
            alert("欢迎!");
        }
        catch(err){
            txt="本页有一个错误。\n\n";
            txt+="错误描述:"+err.message+"\n\n";
            txt+="单击【确定】按钮继续。\n\n";
            alert(txt);
        }
    }
</script>
</head>

<body>
    <input type="button" value="查看消息" onclick="message()" />
</body>
```

由于没有定义 adddlert()这个函数,所以代码出错,处理异常的结果如图 4-14 所示。



图 4-14　异常处理

2. throw

throw 语句允许我们创建自定义错误,如果把 throw 与 try 和 catch 一起使用,那么应能够控制程序流,并生成自定义的错误消息。

格式:

```
throw exception
```

【例 4-21】　自定义异常。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
<script>
    function myFunction(){
        try{
            var x=document.getElementById("demo").value;
            if(x=="")    throw "值为空";
            if(isNaN(x)) throw "不是数字";
            if(x>10)     throw "太大";
            if(x<5)      throw "太小";
        }
        catch(err){
            var y=document.getElementById("mess");
            y.innerHTML="错误："+err+"。";
        }
    }
</script>
<p>请输入 5 到 10 之间的数字:</p>
<input id="demo" type="text">
<button type="button" onclick="myFunction()">测试输入值</button>
<p id="mess"></p>
</head>
</html>
```

结果如图 4-15 所示。

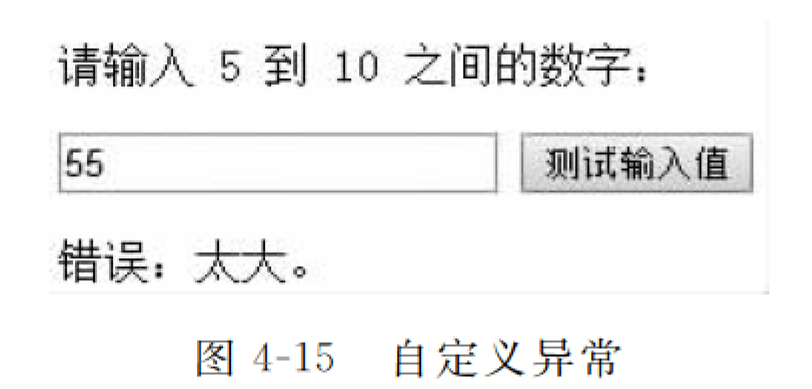


图 4-15　自定义异常

4.6　JavaScript 事件的处理

4.6.1　JavaScript 事件概述

事件是可以被 JavaScript 侦测到的行为，事件可以是浏览器行为也可以是用户行为。与浏览器进行交互的时候，浏览器就会触发各种事件。比如当我们打开某一个网页的时候，浏览器加载完成了这个网页，就会触发一个 load 事件；当我们单击页面中的某一个“地方”，浏览器就会在那个“地方”触发一个 Click 事件。这样，我们就可以编写 JavaScript 代码，通过监听某一个事件来实现某些功能扩展。JavaScript 支持丰富的事件类型，能使 Web 开发更加快速和简洁。

浏览器会根据某些操作触发对应事件，如果我们需要针对某种事件进行处理，则需要监听这个事件。监听事件的方法主要有以下几种。

1. HTML 内联属性(避免使用)

HTML 元素里面直接填写事件的有关属性,属性值为 JavaScript 代码,即可在触发该事件的时候执行属性值的内容。

【例 4-22】 内联属性监听事件。

```
<body>
    <button onclick="alert("你单击了这个按钮");">单击这个按钮</button>
</body>
```

显而易见,使用这种方法,JavaScript 代码与 HTML 代码耦合在了一起,不便于维护和开发。所以除非在必须使用的情况(例如统计链接单击数据)下,尽量避免使用这种方法。

2. DOM 属性绑定

直接设置 DOM 属性来指定某个事件对应的处理函数。

```
element.onclick=function(event){
    alert("你单击了这个按钮");
}
```

3. 使用事件监听函数

标准的事件监听函数如下:

```
element.addEventListener(<event-name>,<callback>,<use-capture>);
```

表示在 element 这个对象上面添加一个事件监听器,当监听到有<event-name>事件发生的时候,调用<callback>这个回调函数。<use-capture>表示该事件监听是在“捕获”阶段中监听(设置为 true)还是在“冒泡”阶段中监听(设置为 false)。

将上面的例子改写成例 4-23。

【例 4-23】 使用事件监听函数监听事件。

```
var btn=document.getElementById("btn");
btn.addEventListener("click", function(){
    alert("你单击了这里");
},false);
```

JavaScript 所支持的事件,可以分为以下几类。

(1) 窗口事件(Window Events)。

(2) 表单元素事件(Form Element Events)。

(3) 键盘事件(Keyboard Events)。

(4) 鼠标事件(Mouse Events)。

(5) 图像事件(Image Events)。

4.6.2 窗口事件

仅在 body 和 frameset 元素中有效。onload 和 onunload 事件会在用户进入或离开页面时被触发。窗口事件的简单说明如表 4-6 所示。

表 4-6 窗口事件的简单说明

事　件	说　　明
onload	当网页被载入时执行脚本
onunload	当网页被关闭时执行脚本

【例 4-24】 网页被载入时提示“网页加载完成”。

```
<body onload="showMessage()">
<script type="text/javascript">
    function showMessage(){
        alert("页面加载完成");
    }
</script>
</body>
```

网页加载的结果如图 4-16 所示，并弹出提示框。



图 4-16 网页加载的结果

4.6.3 表单元素事件

表单元素事件仅在表单元素中有效。表单事件的简单说明如表 4-7 所示。

表 4-7 表单事件的简单说明

事　件	说　　明
onchange	当元素(select、复选框等)改变时执行脚本
onsubmit	当表单(form)被提交时执行脚本
onreset	当表单被重置时执行脚本
onselect	当元素被选取时执行脚本
onblur	当元素失去焦点时执行脚本
onfocus	当元素获得焦点时执行脚本

【例 4-25】 将获得焦点的文本框背景色改为灰色。

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
```

```
<script type="text/javascript">
function getFocus(x){
    x.style.background="red";
}
</script>
<form>
    <input type="text" value="单击获得焦点"  onfocus="getFocus(this)"/>
</form>
</body>
</html>
```

执行以上代码，单击文本框后的结果如图 4-17 所示，背景变为灰色。

单击获得焦点

图 4-17　单击文本框的结果

4.6.4　键盘事件

键盘事件在下列 HTML 元素中无效：base、bdo、br、frame、frameset、head、html、iframe、meta、param、script、style 以及 title。键盘事件的简单说明如表 4-8 所示。

表 4-8　键盘事件的简单说明

事　件	说　　明
onkeydown	当键盘被按下时执行脚本
onkeypress	当键盘被按下后又松开时执行脚本
onkeyup	当键盘被松开时执行脚本

【例 4-26】 按下键盘，文本框背景色改为灰色。

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
<script type="text/javascript">
    function pressKey(x){
        x.style.background="gray";
    }
</script>
<form><input type="text" value="按下键盘"  onkeydown="pressKey(this)"/>
</form>
</body>
</html>
```

执行以上代码，单击选择文本框，按下键盘上的任意键，结果如图 4-18 所示，背景变为灰色。

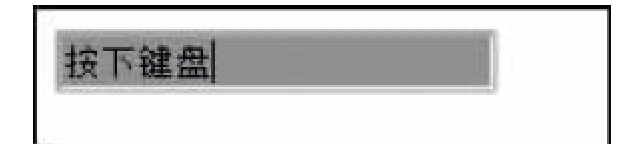


图 4-18　按下键盘上的任意键的运行程序的结果

4.6.5　鼠标事件

鼠标事件在下列 HTML 元素中无效：base、bdo、br、frame、frameset、head、html、iframe、meta、param、script、style 以及 title。鼠标事件的简单说明如表 4-9 所示。

表 4-9　鼠标事件的简单说明

事　件	说　明
onclick	当鼠标被单击时执行脚本
ondblclick	当鼠标被双击时执行脚本
onmousedown	当鼠标按钮被按下时执行脚本
onmousemove	当鼠标指针移动时执行脚本
onmouseout	当鼠标指针移出某元素时执行脚本
onmouseover	当鼠标指针悬停于某元素之上时执行脚本
onmouseup	当鼠标按钮被松开时执行脚本

【例 4-27】　双击鼠标改变 div 背景色。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
<style  type="text/css">
    div{
        width:200px;
        height:100px;
        background-color:#066;
    }
</style>
</head>
<body>
<script type="text/javascript">
    function dbclick(x){
        x.style.background="gray";
    }
</script>
<div  ondblclick="dbclick(this)">请双击</div>
</body>
</html>
```

执行以上代码，双击蓝色区域，结果如图 4-19 所示，背景变为灰色。



图 4-19　双击蓝色区域的结果

4.6.6　图像事件

图像事件可用于 img 元素，图像事件的简单说明如表 4-10 所示。

表 4-10　图像事件的简单说明

事　件	说　　明
onabort	当图像加载中断时执行脚本

4.7　JavaScript DOM

4.7.1　JavaScript HTML DOM 概述

对象文档模型(document object model，DOM)，是 W3C 组织推荐的处理可扩展置标语言的标准编程接口。DOM 可以以一种独立于平台和语言的方式访问和修改一个文档的内容和结构。也就是说，这是表示和处理一个 HTML 或 XML 文档的常用方法。DOM 技术使用户页面可以动态地变化，从而使页面的交互性大大增强。但是 DOM 必须通过 JavaScript 等脚本语言来进行读取，可以改变 HTML、XHTML 以及 XML 等文档。简单来说就是 DOM 规定了 HTML、XML 等的一些规范，使 JavaScript 可以根据这些规范来进行各种操作。这些规范可以用树状图来形象地表示，如图 4-20 所示。

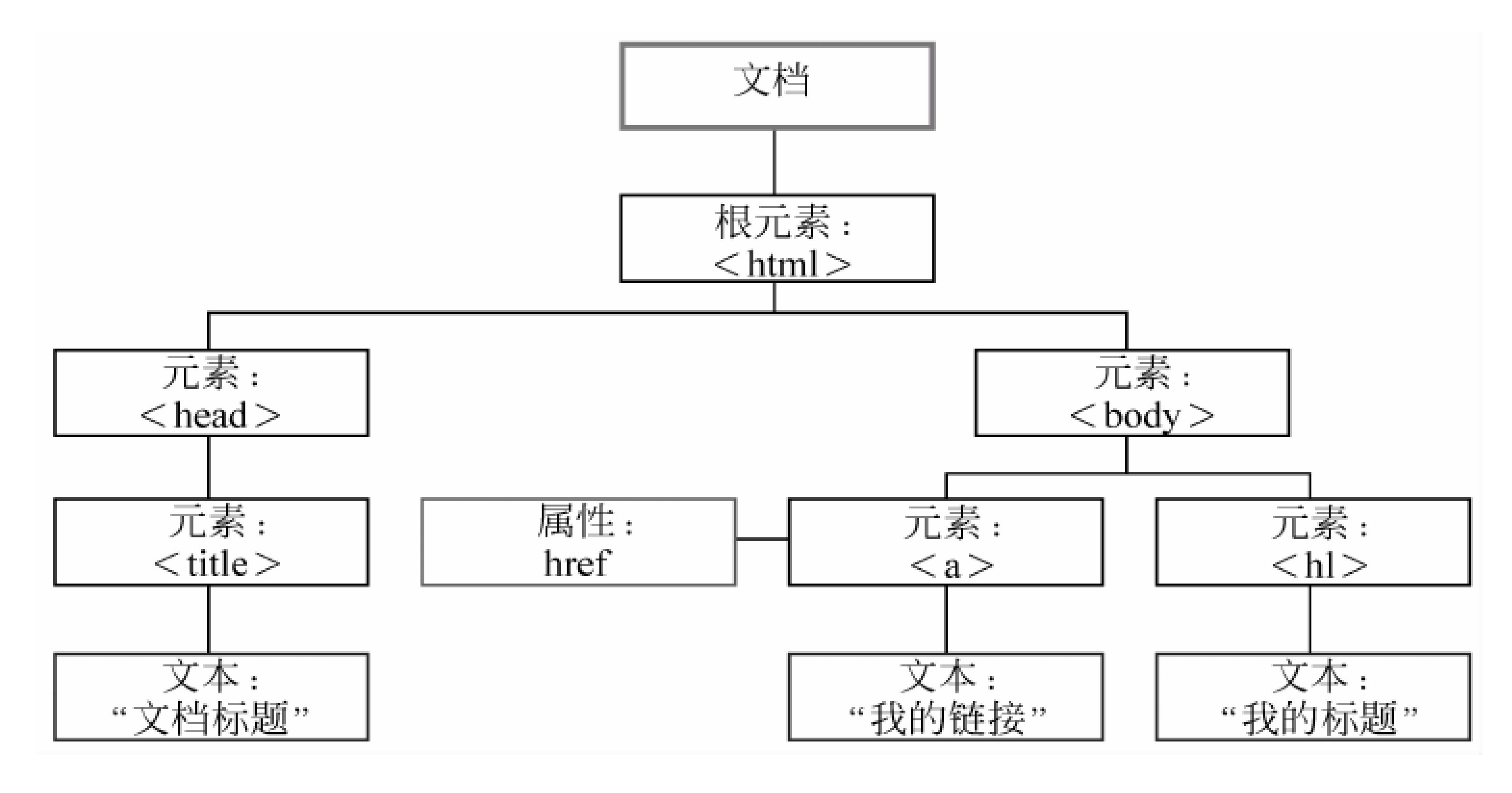


图 4-20　DOM 树

通过这样的树，就可以很快找到我们想要操作的节点，进而进行各种属性、方法、事件等的操作。通过可编程的对象模型，JavaScript 获得了足够的能力来创建动态的 HTML。

- JavaScript 能够改变页面中的所有 HTML 元素。
- JavaScript 能够改变页面中的所有 HTML 属性。
- JavaScript 能够改变页面中的所有 CSS 样式。
- JavaScript 能够对页面中的所有事件做出反应。

4.7.2　HTML DOM 对象

HTML DOM 定义了访问和操作 HTML 文档的标准方法。它把 HTML 文档呈现为带有元素、属性和文本的树结构。在层次图中，每个对象是它的父对象的属性，如 Window 对象是 Document 对象的父对象，所以再引用 Document 对象就可以使用 Window. Document，相当于 Document 是 Window 对象的属性。对于每一个页面，浏览器都会自动创建 Window 对象、Navigator 对象、Screen 对象、History 对象、Location 对象、Document 对象。

1. Window 对象

Window 对象表示浏览器中打开的窗口。如果文档包含框架(frame 或 iframe 标签)，浏览器会为 HTML 文档创建一个 Window 对象，并为每个框架创建一个额外的 Window 对象。Window 对象表示浏览器中打开的窗口，如果文档包含 frame 或 iframe 标签，浏览器会为 HTML 文档创建一个 Window 对象，并为每个框架创建一个额外的 Window 对象。在客户端 JavaScript 中，Window 对象是全局对象，所有的表达式都在当前环境中计算，因此可以把窗口的属性作为全局变量来使用，例如可以只写 doucment，而不必写 Window. document。同样可以把窗口对象的方法当成函数使用，如只写 alert()，而不必写 Window. alert()。Window 对象的属性及方法如表 4-11 所示。

表 4-11　Window 对象的属性及方法

属性/方法	描　　述
close 属性	返回一个布尔值，声明了窗口是否已经关闭，为只读属性
defaultStatus 属性	设置或返回窗口状态栏中的默认文本，为只读属性
opener 属性	返回创建该窗口的 Window 对象的引用，是一个可读可写的属性
self 属性	返回窗口自身的引用，相当于 Window 属性
doucment 属性	对 Document 对象的只读引用
alert()方法	用于显示带有一条指定消息和一个 OK 按钮的警告框
confirm()方法	用于显示一个带有指定消息、OK 及 Cancel 按钮的对话框
prompt()方法	用于显示可使用户进行输入的对话框
createPopup()方法	用于创建一个 pop-up 窗口
setInterval()方法	按照指定的周期(以毫秒计)来调用函数或计算表达式。它会不停地调用函数，直到 clearInterval()被调用或窗口被关闭
clearInterval()方法	取消由 setInterval()设置的 timeout
setTimeout()方法	用于在指定的毫秒数后调用函数或计算表达式
write()方法	向文档写 HTML 表达式或 JavaScript 代码
writeln()方法	等同于 write()方法，不同的是在每个表达式之后写一个换行符

【例 4-28】 Window 对象方法的应用。

```
<html>
<head>
<script language=javascript>
    function display_alert() {
        alert("I am an alert box!")
    }
    function display_confirm() {
        var r=confirm("press a button")
        if(r==true)   {
            document.write("You pressed OK!")
        }
        else  {
            document.write("You pressed Cancel!")
        }
    }
    function display_prompt() {
        var name=prompt("please enter your name","")
        if(name!=null&&name!="")
        {
            document.write("Hello "+name+"!")
        }
    }
</script>
</head>
<body>
    <input type="button" onclick="display_alert()" value="Display alert
    box"/>
    <br/>
    <input type="button" onclick="display_confirm()" value="Display a confirm
    box"/>
    <br/>
    <input type="button" onclick="display_prompt()" value="Display a prompt
    box"/>
</body>
</html>
```

2. Navigator 对象

Navigator 对象包含有关浏览器的信息，Navigator 对象包含的属性描述了正在使用的浏览器，可以使用这些属性进行平台的专用配置，这个实例是唯一的，可以通过 window.navigator 属性来应用它。Navigator 对象常用的属性和方法如表 4-12 所示。

表 4-12 Navigator 对象常用的属性和方法

属性/方法	描　述
appCodeName 属性	返回浏览器的代码名
appMinorVersion 属性	返回浏览器的次级版本
appName 属性	返回浏览器的名称

续表

属性/方法	描　　述
appVersion 属性	返回浏览器的平台和版本信息。返回窗口自身的引用，相当于 Window 属性
browserLanguage 属性	返回当前浏览器的语言
cookieEnabled 属性	返回指明浏览器中是否启用 cookie 的布尔值
javaEnabled()方法	规定浏览器是否启用 Java
taintEnabled()方法	规定浏览器是否启用数据污点（data tainting）

3. Screen 对象

Screen 对象包含有关客户端显示屏幕的信息。每个 Window 对象的 screen 属性都引用一个 Screen 对象。Screen 对象中存放着有关显示浏览器屏幕的信息。JavaScript 程序将利用这些信息来优化它们的输出，以达到用户的显示要求。例如，一个程序可以根据显示器的尺寸选择使用大图像还是使用小图像，它还可以根据显示器的颜色深度选择使用 16 位色还是使用 8 位色的图形。另外，JavaScript 程序还能根据有关屏幕尺寸的信息将新的浏览器窗口定位在屏幕中间。Screen 对象的常用属性如表 4-13 所示。

表 4-13　Screen 对象的常用属性

属　性	描　　述
availHeight	返回显示屏幕的高度(除 Windows 任务栏之外)
availWidth	返回显示屏幕的宽度(除 Windows 任务栏之外)
bufferDepth	设置或返回调色板的比特深度
colorDepth	返回目标设备或缓冲器上的调色板的比特深度
deviceXDPI	返回显示屏幕的每英寸水平点数
deviceYDPI	返回显示屏幕的每英寸垂直点数
height	返回显示屏幕的高度
pixelDepth	返回显示屏幕的颜色分辨率(比特每像素)

4. History 对象

History 对象包含用户(在浏览器窗口中)访问过的 URL。History 对象是 Window 对象的一部分，可通过 Window. history 属性对其进行访问。History 对象常用的属性及方法如表 4-14 所示。

表 4-14　History 对象常用的属性及方法

属性/方法	描　　述
length 属性	返回浏览器历史列表中的 URL 数量
back()方法	加载历史列表中的前一个 URL
forward()方法	加载历史列表中的下一个 URL
go()方法	加载历史列表中的某个具体页面

5. Location 对象

Location 对象包含有关当前 URL 的信息。Location 对象是 Window 对象的一个部分，可通过 Window. location 属性来访问。Location 对象常用的属性及方法如表 4-15 所示。

表 4-15　Location 对象常用的属性及方法

属性/方法	描　　述
hash 属性	设置或返回从井号(#)开始的 URL(锚)
hostname 属性	设置或返回当前 URL 的主机名
href 属性	设置或返回完整的 URL
pathname 属性	设置或返回当前 URL 的路径部分
assign()方法	加载新的文档
reload()方法	重新加载当前文档
replace()方法	用新的文档替换当前文档

6. Document 对象

每个载入浏览器的 HTML 文档都会成为 Document 对象。Document 对象使我们可以从脚本中对 HTML 页面中的所有元素进行访问。Document 对象的属性及方法如表 4-16 所示。

表 4-16　Document 对象的属性及方法

属性/方法	描　　述
body 属性	提供对<body>元素的直接访问。对于定义了框架集的文档，该属性引用最外层的框架集
cookie 属性	设置或返回与当前文档有关的所有 cookie
domain 属性	返回当前文档的域名
lastModified 属性	返回文档被最后修改的日期和时间
referrer 属性	返回载入当前文档的 URL
title 属性	返回当前文档的标题
URL 属性	返回当前文档的 URL
open()方法	打开一个新的文档，并擦除当前文档的内容
close()方法	关闭一个由 document. open()方法打开的输出流，并显示选定的数据
getElementById()方法	返回对拥有指定 ID 的第一个对象的引用
getElementsByName()方法	返回带有指定名称对象的集合
getElementsByTagName()方法	返回带有指定标签名的对象的集合
write()方法	向文档写入 HTML 表达式或 JavaScript 代码
writeln()方法	等同于 write()方法，不同的是在每个表达式之后写一个换行符

【例 4-29】　单击“提交”按钮，获取按钮的值。

```
<body>
<script type="text/javascript">
    function getValue(){
        var x=document.getElementById("btn");
        alert(x.value);
    }
</script>
<form>
```

```
    <input type="button" value="提交" onclick="getValue()" id="btn"/>
</form>
</body>
```

执行以上代码，单击“提交”按钮，程序的运行结果如图 4-21 所示。

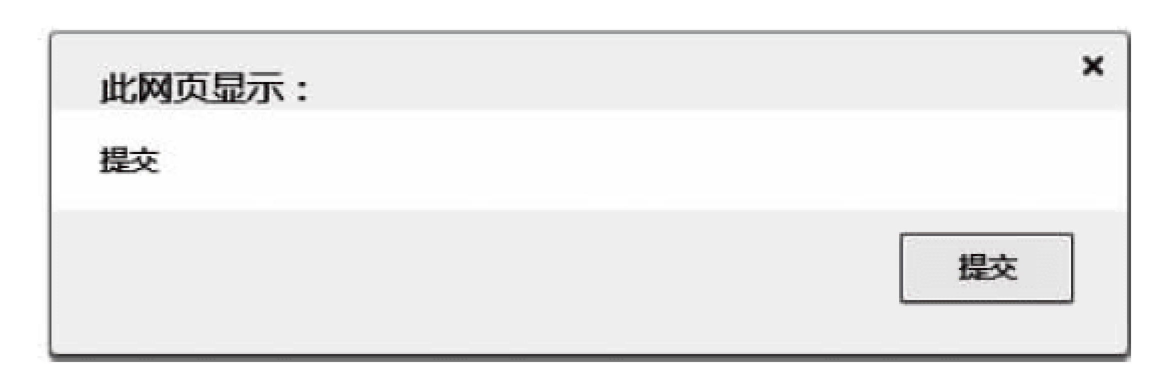


图 4-21　获取“提交”按钮的值

7. Element 对象

在 HTML DOM 中，Element 对象表示 HTML 元素。Element 对象可以拥有类型为元素节点、文本节点、注释节点的子节点。NodeList 对象表示节点列表，比如 HTML 元素的子节点集合。Element 对象的部分属性及方法如表 4-17 所示。

表 4-17　Element 对象的部分属性及方法

属性/方法	描　　述
element. attributes 属性	返回一个元素的属性数组
element. childNodes 属性	返回元素的一个子节点的数组
element. contentEditable 属性	设置或返回元素的内容是否可编辑
element. innerHTML 属性	设置或者返回元素的内容
nodelist. length 属性	返回节点列表的节点数目
element. addEventListener()方法	向指定元素添加事件句柄
element. appendChild()方法	为元素添加一个新的子元素
element. getAttribute()方法	返回指定元素的属性值
element. getAttributeNode()方法	返回指定属性节点
element. getElementsByTagName()方法	返回指定标签名的所有子元素集合
element. getElementsByClassName()方法	返回文档中所有指定类名的元素集合，作为 NodeList 对象
element. removeChild()方法	删除一个子元素
element. setAttribute()方法	设置或者改变指定属性并指定值
nodelist. item()方法	返回某个元素基于文档树的索引

【例 4-30】　单击“单击修改文本”按钮，改变超链接的属性和文本。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script>
    function changeLink(){
        document.getElementById('mya').innerHTML="搜狐";
        document.getElementById('mya').href="http://tv.sohu.com/";
```

```
        document.getElementById('mya').target="_blank";
    }
</script>
</head>
<body>
    <a id="mya" href="http://www.baidu.com">百度</a>
    <input type="button" onclick="changeLink()" value="单击修改文本">
</body>
</html>
```

执行以上代码，结果如图 4-22 所示。

图 4-22　改变超链接的属性和文本

8. Attr 对象

在 HTML DOM 中，Attr 对象表示 HTML 属性。HTML 属性始终属于 HTML 元素。NamedNodeMap 对象表示元素属性节点的无序集合。NamedNodeMap 中的节点可通过名称或索引（数字）来访问。Attr 对象的部分属性及方法如表 4-18 所示。

表 4-18　Attr 对象的部分属性及方法

属性/方法	描　述
attr. isId 属性	如果属性是 id 类型，则返回 true，否则返回 false
attr. name 属性	返回属性的名称
attr. value 属性	设置或返回属性的值
attr. specified 属性	如果已指定属性，则返回 true，否则返回 false
nodemap. length 属性	返回 NamedNodeMap 中的节点数
nodemap. getNamedItem()方法	从 NamedNodeMap 返回指定的属性节点
nodemap. item()方法	返回 NamedNodeMap 中位于指定下标的节点
nodemap. removeNamedItem()方法	移除指定的属性节点
nodemap. setNamedItem()方法	设置指定的属性节点（通过名称）

【例 4-31】　改变标题的大小。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<style type="text/css">
    .testdemo{
        font-size:36px;
    }
</style>
</head>
<body>
    <h1 class="test">字体将变大</h1>
    <button onclick="myFunction()">试一下</button>
<script>
    function myFunction()
```

```
    {
        var h=document.getElementsByTagName("H1")[0];
        var typ=document.createAttribute("class");
        typ.nodeValue="testdemo";
        h.attributes.setNamedItem(typ);
    }
</script>
</body>
</html>
```

执行以上代码，单击“试一下”按钮，标题变大，如图 4-23 所示。

字体将变大

试一下

图 4-23 标题变大

4.7.3 HTML DOM 的简单应用

1. 改变 HTML

【例 4-32】 改变 HTML 输出流。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script>
    document.write(Date());
</script>

</head>
<body>
    <P>这是一个网页</P>
</body>
</html>
```

以上代码除了在网页上显示“这是一个网页”之外，还通过 document.write(Date())在网页中显示了当天的日期，如图 4-24 所示。

Wed Dec 14 2016 13:52:53 GMT+0800 (中国标准时间)

这是一个网页

图 4-24 改变 HTML 输出流

【例 4-33】 改变 HTML 的内容。

```
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="text/javascript">
    document.getElementById("p1").innerHTML="新文本!";
</script>
</head>
<body>
    <p id="p1">段落文本</p>
</body>
</html>
```

执行以上代码，本来网页中应该显示“段落文本”，但是通过代码“document.getElementById("p1").innerHTML="新文本!"”将段落文本更改为“新文本”。

【例 4-34】 改变 HTML 属性。

```
<html>
<head>
<script>
    document.getElementById("image").src="landscape.jpg";
</script>
</head>
<body>
    <img id="image" src="smiley.gif">
</body>
</html>
```

执行以上代码，通过代码“document.getElementById("image").src="landscape.jpg";”将原来图片的 src 属性的值更改为“landscape.jpg”。

2. 改变 CSS

HTML DOM 允许 JavaScript 改变 HTML 元素的样式。

语法如下：

```
document.getElementById(id).style.property=新样式
```

【例 4-35】 改变 HTML 样式。

```
<html>
<head>
<meta charset="utf-8">
<title></title>
</head>
<body>
    <p id="p1">Hello World!</p>
    <p id="p2">Hello World!</p>
<script>
    document.getElementById("p2").style.color="blue";
    document.getElementById("p2").style.fontFamily="Arial";
    document.getElementById("p2").style.fontSize="larger";
```

```
</script>
    <p>以上段落通过脚本修改。</p>
</body>
</html>
```

通过 document 对象将段落 id 为 p2 的段落的颜色设置为蓝色，将字体设置为 Arial，将字体大小设置为 larger。对比结果如图 4-25 所示。

Hello World!

Hello World!

以上段落通过脚本修改。

图 4-25　改变 HTML 样式

如需向 HTML DOM 添加新元素，则必须首先创建该元素（元素节点），然后向一个已存在的元素追加该元素。

【例 4-36】　创建新的 HTML 元素。

```
<div id="div1">
    <p id="p1">这是一个段落。</p>
    <p id="p2">这是另一个段落。</p>
</div>
<script>
    var para=document.createElement("p");
    var node=document.createTextNode("这是一个新段落。");
    para.appendChild(node);
    var element=document.getElementById("div1");
    element.appendChild(para);
</script>
```

以上代码通过 document. createElement 方法创建一个新元素 p，并将新元素追加到已存在的元素 div 上。

【例 4-37】　删除一个段落。

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
    <div id="div1">
        <p id="p1">这是一个段落。</p>
        <p id="p2">这是另一个段落。</p>
    </div>
<script>
    var parent=document.getElementById("div1");
    var child=document.getElementById("p1");
    parent.removeChild(child);
</script>
</body>
</html>
```

以上代码删除段落 p1。

4.8 综合实例

本节我们将通过一个案例来综合应用 JavaScript，帮助大家掌握并理解 JavaScript 的基本语法和基本应用。

4.8.1 需求描述

创建一个表单，表单样式如图 4-26 所示，再对表单进行验证，验证通过则提示提交成功。验证要求如下：

（1）用户名不能为空。

（2）“确认密码”与“密码”两个文本框的输入值必须一致且不能为空。

（3）E-mail 必须符合邮箱规则。

（4）文本框失去焦点时进行验证。

（5）所有验证通过单击“注册”按钮并提示“success”，否则显示未通过的文本框。

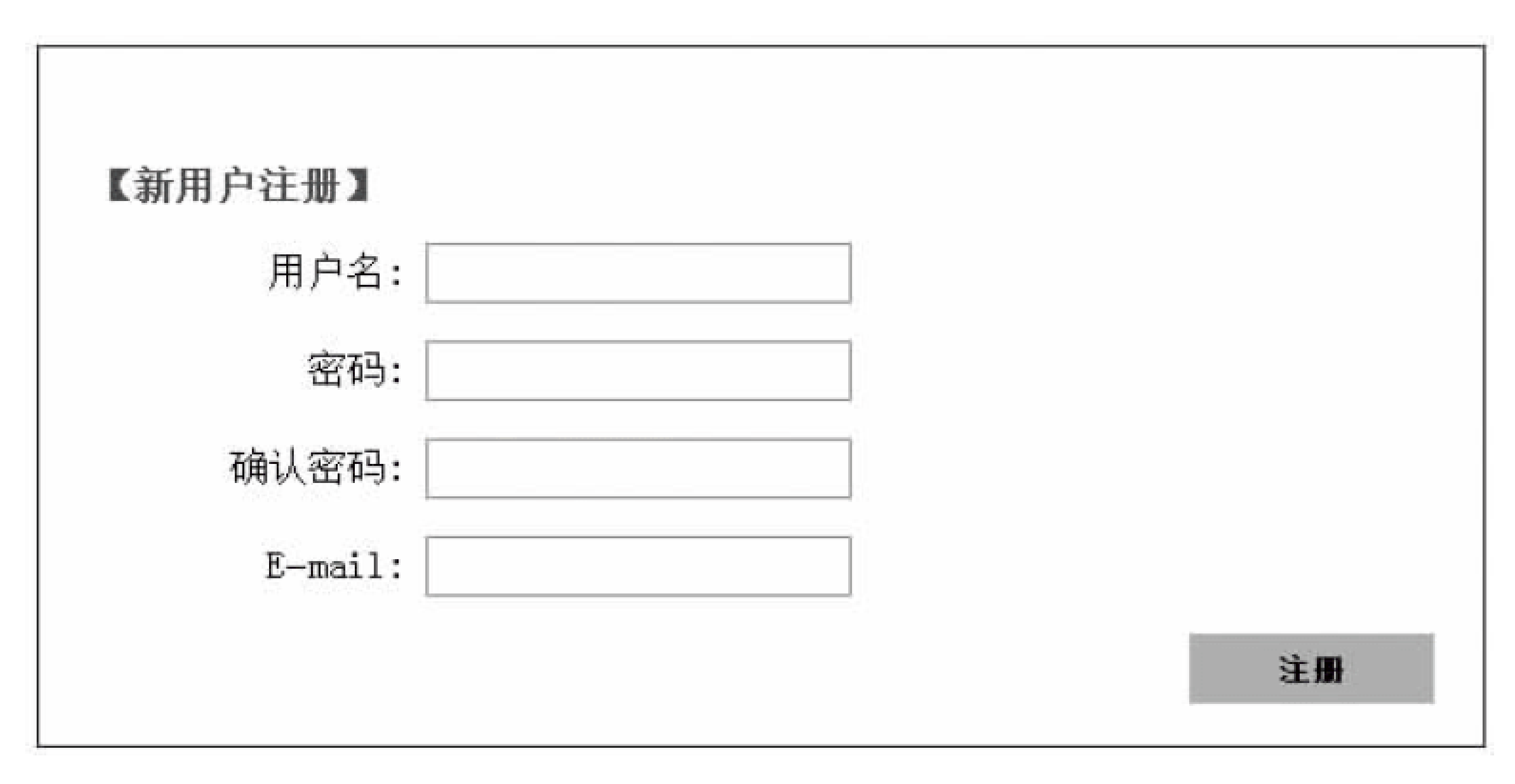


图 4-26　表单样式

4.8.2 分析及实现

1. 分析

要按以上要求进行数据验证，则需实现如下功能。

（1）单击“提交”按钮不能直接提交数据，应该调用一个函数来验证数据。

（2）验证数据必须获取文本框的值，并判断值的有效性。

（3）验证未通过时样式如图 4-27 所示。

2. 实现代码

（1）网页内容。

```
<div class="login">
    <form name="form" method="post" action="register.php" onsubmit="return
    check()">
```

【新用户注册】
用户名: *不能为空
密码:
确认密码: *不能为空且必须一致
E-mail: *格式不正确
注册

图 4-27　验证未通过的样式

```
    <legend>【新用户注册】</legend>
        <p><label for="name">用户名:</label>
            <input type="text" id="name"><label id="cname"> * 不能为空
            </label>
        </p>
        <p><label for="password">密码:</label>
            <input type="password" id="password">
            <label id="cpassword"> * 不能为空且必须一致</label>
        </p>
        <p><label for="R_password">确认密码:</label>
            <input type="password" id="R_password">
            <label id="crepass"> * 不能为空且必须一致</label>
        </p>
        <p>
            <label for="email">E-mail:</label>
            <input type="text" id="email">
            <label id="cemail"> * 格式不正确</label>
        </p>
        <p>
            <input type="submit" value="注册" class="btn">
        </p>
    </form>
</div>
```

(2) CSS 样式。

```
<style type="text/css">
    body{margin:0;
        padding: 0;
    }
    .login{
        position:relative;
        margin:100px auto;
        padding:50px 20px;
```

```
        width: 550px;
        height: 200px;
        border:1px solid #333;
    }
    .login legend{
        font-weight: bold;
        color: green;
        text-align: center;
    }
    .login label{
        display:inline-block;
        width:130px;
        text-align: right;
    }
    .btn{
        height: 30px;
        width:100px;
        padding: 5px;
        border:0;
        background-color: #00dddd;
        font-weight: bold;
        cursor: pointer;
        float: right;
    }
    input{
        height: 20px;
        width: 170px;
    }
    .borderRed{
        border: 2px solid red;
    }
    #cname,#cpassword,#crepass,#cemail{
        display:none;
        color:#F00;
    }
</style>
```

(3）完整的 JavaScript 代码。

```
<script type="text/javascript">
  function hasClass(obj,cls){                    //判断 obj 是否有此类
    return obj.className.match(new RegExp('(\\s|^)'+cls+'(\\s|$)'));
  }

  function addClass(obj,cls){                    //给 obj 添加类
    if(!this.hasClass(obj,cls)){
      obj.className+=" "+cls;
    }
  }
```

```
function removeClass(obj,cls){                    //移除 obj 对应的类
  if(hasClass(obj,cls)){
    var reg=new RegExp('(\\s|^)'+cls+'(\\s|$)');
    obj.className=obj.className.replace(reg," ");
  }
}

function checkName(name){                         //验证 name
  if(name !=""){  //不为空则正确。当然也可以用 ajax 异步获取,用户名不重复则正确
    removeClass(ele.name,"borderRed");            //移除类
    document.getElementById('cname').innerHTML='* 正确';
    document.getElementById("cname").style.display="inline";
    return true;
  }else{  //name 不符合
    addClass(ele.name,"borderRed");               //添加类
     document.getElementById("cname").style.display="inline";
    return false;
  }
}

function checkPassw(passw1,passw2){               //验证密码
  if(passw1=="" || passw2=="" || passw1 !==passw2){
                               //两次输入的密码有一次为空或两者不相等,则不符合要求
    addClass(ele.password,"borderRed");
    addClass(ele.R_password,"borderRed");
    document.getElementById("crepass").style.display="inline";
    return false;
  }else{                                          //密码输入正确
    removeClass(ele.password,"borderRed");
    removeClass(ele.R_password,"borderRed");
    document.getElementById('cpassword').innerHTML='* 正确';
    document.getElementById("cpassword").style.display="inline";
    document.getElementById('crepass').innerHTML='* 正确';
    document.getElementById("crepass").style.display="inline";
    return true;
  }
}
function checkEmail(email){                       //验证邮箱
  var pattern=/^([\.a-zA-Z0-9_-])+@([a-zA-Z0-9_-])+(\.[a-zA-Z0-9_-])+/;
  if(!pattern.test(email)){                       //email 格式不正确
    addClass(ele.email,"borderRed");
    document.getElementById("cemail").style.display="inline";
    ele.email.select();
    return false;
  }else{                                          //格式正确
    removeClass(ele.email,"borderRed");
    document.getElementById('cemail').innerHTML='* 正确';
    document.getElementById("cemail").style.display="inline";
```

```
      return true;
    }
  }

    var ele={       //存放各个 input 字段的 obj
      name: document.getElementById("name"),
      password: document.getElementById("password"),
      R_password: document.getElementById("R_password"),
      email: document.getElementById("email")
    };
    ele.name.onblur=function(){           //name 失去焦点则进行检测
      checkName(ele.name.value);
    }
    ele.password.onblur=function(){       //password 失去焦点则进行检测
      checkPassw(ele.password.value,ele.R_password.value);
    }
    ele.R_password.onblur=function(){     //R_password 失去焦点则进行检测
      checkPassw(ele.password.value,ele.R_password.value);
    }
    ele.email.onblur=function(){          //email 失去焦点则进行检测
      checkEmail(ele.email.value);
    }

    //表单提交时则验证函数
  function check(){
    var ok=false;
    var nameOk=false;
    var emailOk=false;
    var passwOk=false;

    //验证 name
    if(checkName(ele.name.value)){ nameOk=true; }
    //验证 password
    if(checkPassw(ele.password.value,ele.R_password.value))
    {
        passwOk=true;
    }

    //验证电子邮件
    if(checkEmail(ele.email.value))
    {
        emailOk=true;
    }

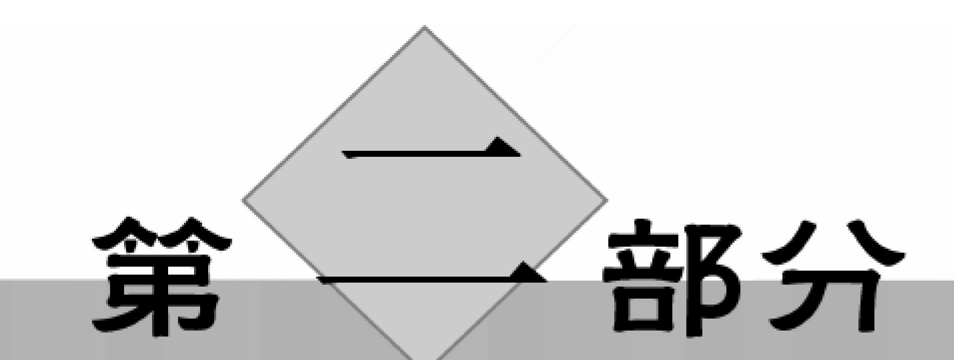
    //注册成功
    if(nameOk && passwOk && emailOk){
        alert("Tip: Register Success ..");
        //return true;
    }
    return false;  //有错误,注册失败
  }
</script>
```

本章小结

JavaScript 是一种属于网络的脚本语言，已经被广泛用于 Web 应用开发，常用来为网页添加各式各样的动态功能，为用户提供更流畅美观的浏览效果。

本章从 JavaScript 的概念出发，对 JavaScript 的特点、优点、缺陷等方面做了详细介绍，重点介绍了 JavaScript 的语法。JavaScript 的语法丰富，简单易学。JavaScript 运算符包含算术运算符、赋值运算符、比较运算符、逻辑运算符、条件运算符等。JavaScript 函数也有返回值和参数。HTML DOM 符合 W3C 标准。HTML DOM 定义了用于 HTML 的一系列标准的对象，以及访问和处理 HTML 文档的标准方法。通过 DOM 可以访问所有的 HTML 元素，连同它们所包含的文本和属性。可以对其中的内容进行修改和删除，同时也可以创建新的元素。





进　阶　篇



第 5 章　JavaScript 高级编程

5.1　面向过程编程和面向对象编程概述

5.1.1　面向过程编程

1. 什么是面向过程

面向过程编程是一种以过程为中心的编程思想，先分析出解决问题的步骤，然后用函数把这些步骤一步一步实现。进行面向过程的编程时，数据和对数据的操作是分离的。一般的面向过程是从上往下步步求精，所以面向过程最重要的是模块化的思想方法。比如，学生早上起来粗略地可以将过程拟定为：

(1) 起床

(2) 穿衣

(3) 洗脸及刷牙

(4) 去学校

而这 4 步就是一步一步地完成的，它的顺序很重要，只需一项一项地实现就可以了。

2. 面向过程编程的优缺点

面向过程编程的优点：首先，程序顺序执行，流程清晰明了。其次，面向过程的性能比面向对象要高，因为类调用时需要实例化，开销比较大，比较消耗资源；比如单片机、嵌入式开发、Linux/UNIX 等一般采用面向过程的开发，性能是最重要的因素。所以当我们开发一个不是很复杂的程序，同时对性能方面又有比较高的要求时，面向过程就显得十分高效。

面向过程编程的缺点：主控程序承担任务多，各个模块都需要主控程序进行控制和调度，主控和模块之间承担的任务不均衡；面向过程定义的函数无法方便地进行扩展，重用性低，不像面向对象程序那样易维护、易复用、易扩展。另外，在面向过程的程序中，许多重要的数据被放置在全局数据区，这样它们可以被所有的函数访问，很容易造成全局数据在无意中被其他函数改动，因而程序的正确性不易保证。

5.1.2　面向对象编程

1. 什么是面向对象

面向对象是在结构化设计方法出现很多问题的情况下应运而生的，面向对象程序设计(OOP)技术汲取了结构化程序设计中的好的思想，并将这些思想与一些新的、强大的理念

相结合，从而给程序设计提供了一种全新的方法。通常，在面向对象的程序设计风格中，会将一个问题分解为一些相互关联的子集，每个子集内部都包括了相关的数据和函数。同时，会以某种方法将这些子集分为不同等级，而一个对象就会成为一个已定义的某种类型的变量。当定义了一个对象，就隐含地创建了一个新的数据类型。面向对象编程是将事物对象化，通过对象的通信来解决问题。面向对象编程时，数据和对数据的操作是绑定在一起的。同样，以学生早上起来的过程为例，只需要抽象出一个学生的类，该类同样包括了以上四个方法，但是具体的顺序就不一定按照原来的顺序。

2. 面向对象编程的优缺点

面向对象编程的优点如下：

(1) 易维护。采用面向对象思想设计的结构，可读性高，由于继承的存在，即使改变需求，那么维护也只是限定在局部模块，所以维护起来非常方便，成本也较低。

(2) 质量高。在设计时可重用现有的代码，在以前的项目中已被测试过的类只要满足业务需求并具有较高的质量，即可再次使用。

(3) 效率高。在开发软件时，根据设计的需要对现实世界的事物进行抽象，并产生出类。使用这样的方法解决问题，接近于日常生活和自然的思考方式，势必会提高软件开发的效率和质量。

(4) 易扩展。由于继承、封装、多态的特性，自然设计出高内聚、低耦合的系统结构，使得系统更灵活、更容易扩展，而且成本较低。

面向对象编程的缺点如下：

(1) 需要一定的软件支持环境。

(2) 不太适宜大型的信息系统开发。若缺乏整体系统的设计及划分，易造成系统结构不合理、各部分关系失调等问题。

(3) 只能在现有业务基础上进行分类整理，不能从科学管理角度进行理顺和优化。初学者不易接受，较难学。

(4) 增加工作量。如果一味地强调封装，当进行修改对象内部时，对象的任何属性都不允许外部直接存取，则要增加许多没有其他意义、只负责读或写的行为。这会为编程工作增加负担，增加运行开销，并且使程序显得臃肿。

(5) 性能低。由于面向更高的逻辑抽象层，使得面向对象的程序在实现的时候，不得不做出性能上的牺牲，计算时间和存储空间都开销很大。

3. 面向对象的三大特征

(1) 封装。也就是把客观事物封装成抽象的类，并且类可以把自己的数据和方法只让可信的类或者对象操作，对不可信的进行信息隐藏。封装是面向对象的特征之一，是对象和类概念的主要特性。简单地说，一个类就是一个封装了数据以及操作这些数据的代码的逻辑实体。在一个对象内部，某些代码或某些数据可以是私有的，不能被外界访问。通过这种方式，对象对内部数据提供了不同级别的保护，以防止程序中无关的部分意外地改变或错误地使用了对象的私有部分。

(2) 继承。所谓继承，是指可以让某个类型的对象获得另一个类型的对象的属性和方

法。继承是指这样一种能力：它可以使用现有类的所有功能，并在无须重新编写原来类的情况下对这些功能进行扩展。通过继承创建的新类称为“子类”或“派生类”，被继承的类称为“基类”“父类”或“超类”。继承的过程，就是从一般到特殊的过程。要实现继承，可以通过“继承”(Inheritance)和“组合”(Composition)来实现。继承概念的实现方式有两类：实现继承与接口继承。实现继承是指直接使用基类的属性和方法而无须额外编码的能力；接口继承是指仅使用属性和方法的名称，但是子类必须提供实现的能力。

(3) 多态。多态就是指一个类实例的相同方法在不同情形下有不同的表现形式。多态机制使具有不同内部结构的对象可以共享相同的外部接口。这意味着，虽然针对不同对象的具体操作不同，但通过一个公共的类，它们(那些操作)可以通过相同的方式予以调用。

5.2 JavaScript 的面向对象编程

5.2.1 对象的创建与调用

JavaScript 是面向对象的语言，但 JavaScript 不使用类。在 JavaScript 中不会创建类，也不会通过类来创建对象。JavaScript 中的所有事物都是对象：字符串、数值、数组、函数等。创建新对象有两种不同的方法。

1. 直接定义并创建对象的实例

格式：

```
对象名=new Object();
对象名.属性名=值;
```

创建好对象以后，可以直接通过对象名访问属性和方法，如例 5-1 所示。

【例 5-1】 创建一个 person 对象并赋值。

```
person=new Object();
person.firstname="John";
person.lastname="Doe";
person.age=50;
person.eyecolor="blue";
```

创建好对象后，可以通过 person.age 获取 age 的值。

2. 使用对象构造器创建对象

对象构造器格式：

```
function 构造器名(参数 1,参数 2,...){
    属性=参数 1;
}
```

创建好对象构造器之后就可以创建对象了，如例 5-2 所示。

格式：

```
var 对象名=new 构造器名(值 1,值 2...);
```

创建好对象以后，可以直接通过对象名访问属性和方法。

【例 5-2】 创建一个 person 对象并赋值。

```
function person(firstname,lastname,age,eyecolor){        //对象构造器
    this.firstname=firstname;
    this.lastname=lastname;
    this.age=age;
    this.eyecolor=eyecolor;
}
var myFather=new person("John","Doe",50,"blue");         //创建对象 myFather
var myMother=new person("Sally","Rally",48,"green");     //创建对象 myMother
```

创建好对象后可以通过 myMother. age 获取 age 的值。

5.2.2 常用的内置对象

1. 数字对象

JavaScript 只有一种数字类型。可以使用也可以不使用小数点来书写数字。

例如：

```
var pi=3.14;            //使用小数点
var x=34;               //不使用小数点
```

极大或极小的数字可通过科学(指数)计数法来写：

```
var y=123e5;            //12300000
var z=123e-5;           //0.00123
```

JavaScript 不是类型语言。与许多其他编程语言不同，JavaScript 不定义不同类型的数字，比如整数、短、长、浮点等。在 JavaScript 中，数字不分为整数类型和浮点型类型，所有的数字都是浮点型类型。JavaScript 采用 IEEE 754 标准定义的 64 位浮点格式表示数字，它能表示最大值为$\pm 1.7976931348623157\times 10^{308}$，最小值为$\pm 5\times 10^{-324}$。

关于精度，整数(不使用小数点或指数计数法)最多为 15 位，小数的最大位数是 17 位，但是浮点运算并不总是 100% 准确。

【例 5-3】 小数的精度。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title></title>
</head>
<body>
```

```
<script>
    var x;
    document.write("<p>仅显示 17 位：");
    x=12345678901234567890;
    document.write(x+"</p>");
    document.write("<p>0.2+0.1=");
    x=0.2+0.1;
    document.write(x+"</p>");
    document.write("<p>可以通过乘以 10 或除以 10 来计算：");
    x=(0.2*10+0.1*10)/10;
    document.write(x+"</p>");
</script>
</body>
</html>
```

数字可以直接对数据进行初始化，例如 var x＝123。数字对象初始化数据，例如 var y＝new Number(123)。数字对象常用的属性及方法如表 5-1 所示。

表 5-1　数字对象常用的属性及方法

属性/方法	描　述
constructor 属性	返回对创建此对象的 Number 函数的引用
MAX_VALUE 属性	可表示的最大的数
MIN_VALUE 属性	可表示的最小的数
NaN 属性	非数字值
NEGATIVE_INFINITY 属性	负无穷大，溢出时返回该值
POSITIVE_INFINITY 属性	正无穷大，溢出时返回该值
prototype()方法	使用户有能力向对象添加属性和方法
toString()方法	把数字转换为字符串，使用指定的基数
toLocaleString()方法	把数字转换为字符串，使用本地数字格式顺序
toFixed()方法	把数字转换为字符串，结果的小数点后有指定位数的数字
toExponential()方法	把对象的值转换为指数计数法
toPrecision()方法	把数字格式化为指定的长度
valueOf()方法	返回一个 Number 对象的基本数字值

【例 5-4】　Number 对象的 toFixed 方法。

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
    数字 13.37 只保留一位小数：
<script type="text/javascript">
    var num=new Number(13.37);
    document.write (num.toFixed(1));
```

```
</script>
</body>
</html>
```

以上代码的执行结果如图 5-1 所示。

数字13.37只保留一位小数： 13.4

图 5-1 Number 对象的 toFixed 方法的应用

2. 字符串(String)对象

String 对象用于处理已有的字符块，一个字符串用于存储一系列字符，就像"helloworld"。一个字符串可以使用单引号或双引号。字符串对象常用的属性及方法如表 5-2 所示。

表 5-2 字符串对象常用的属性及方法

属性/方法	描　　述
constructor 属性	对创建该对象的函数的引用
length 属性	字符串的长度
prototype 属性	允许用户向对象添加属性和方法
anchor()方法	创建 HTML 锚
blink()方法	显示闪动字符串
bold()方法	使用粗体显示字符串
charAt()方法	返回在指定位置的字符
charCodeAt()方法	返回在指定位置的字符的 Unicode 编码
concat()方法	连接字符串
fixed()方法	以打字机文本显示字符串
fontcolor()方法	使用指定的颜色来显示字符串
fontsize()方法	使用指定的尺寸来显示字符串
fromCharCode()方法	从字符编码创建一个字符串
indexOf()方法	检索字符串
match()方法	找到一个或多个正则表达式的匹配
split()方法	把字符串分割为字符串数组
substring()方法	提取字符串中两个指定的索引号之间的字符
toUpperCase()方法	把字符串转换为大写

【例 5-5】 使字符串以斜体显示。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="text/javascript">
    var s="how are you";
    document.write( s.italics());
</script>
</head>
<body>
</body>
</html>
```

以上代码执行的结果如图 5-2 所示。

how are you

图 5-2　字符串以斜体显示

3. 日期对象

日期对象用于处理日期和时间。使用 var myDate＝new Date()可创建日期对象。日期对象常用的属性及方法如表 5-3 所示。

表 5-3　日期对象常用的属性及方法

属性/方法	描　　述
constructor 属性	返回对创建此对象的 Date 函数的引用
prototype 属性	允许用户向对象添加属性和方法
date()方法	返回当前的日期和时间
getDate()方法	从 Date 对象返回一个月中的某一天(1～31)
getDay()方法	从 Date 对象返回一周中的某一天(0～6)
getHours()方法	返回 Date 对象的小时(0～23)
setDate()方法	设置 Date 对象中月的某一天(1～31)
setMonth()方法	设置 Date 对象中的月份(0～11)
setMinutes()方法	设置 Date 对象中的分钟(0～59)
toString()方法	把 Date 对象转换为字符串

【例 5-6】　输出当前的年月日。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="text/javascript">
    var d=new Date();
    var txt="今天是"+d.getFullYear()+"年"+(d.getMonth()+1)+"月"+d.getDate()+"
            日";
    document.write(txt);
</script>
</head>
<body>
</body>
</html>
```

以上代码的执行结果如图 5-3 所示。

今天是2016年12月7日

图 5-3　输出当前的年月日

4. Array(数组)对象

数组对象的作用是：使用单独的变量名来存储一系列的值。数组对象常用的属性及方法如表 5-4 所示。

表 5-4　数组对象常用的属性及方法

属性/方法	描　　述
constructor 属性	返回对创建此对象的 Date 函数的引用
prototype 属性	允许用户向对象添加属性和方法
length 属性	设置或返回数组中元素的数目
concat()方法	连接两个或更多的数组，并返回结果
join()方法	把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔
sort()方法	对数组的元素进行排序
splice()方法	删除元素，并向数组添加新元素
toString()方法	把数组转换为字符串，并返回结果
setMinutes()方法	设置 Date 对象中的分钟(0～59)
toString()方法	把 Date 对象转换为字符串

【例 5-7】 用逗号连接数组的元素并使之成为字符串。

```
<script type="text/javascript">
    var arr=new Array(3);
    arr[0]="how";
    arr[1]="are";
    arr[2]="you";
    var s=arr.join(",");
    document.write(s);
</script>
```

5. Boolean(布尔)对象

Boolean(布尔)对象用于将非布尔值转换为布尔值(true 或者 false)。布尔对象常用的属性及方法如表 5-5 所示。

表 5-5　布尔对象常用的属性及方法

属性/方法	描　　述
constructor 属性	返回对创建此对象的 Boolean 函数的引用
prototype 属性	允许用户向对象添加属性和方法
toString()方法	把 Date 对象转换为字符串
valueOf()方法	返回 Boolean 对象的原始值

【例 5-8】 将布尔值转换为字符串。

```
var bool=new Boolean(1);
var myvar=bool.toString();
```

6. Math 对象

Math 对象的作用是执行数学计算，它提供了标准的数学常量和函数库。Math 对象常用的属性及方法如表 5-6 所示。

表 5-6　Math 对象常用的属性及方法

属性/方法	描　　述
E 属性	返回算术常量 e，即自然对数的底数(约等于 2.718)
LN2 属性	返回 2 的自然对数(约等于 0.693)
LOG2E 属性	返回以 2 为底的 e 的对数(约等于 1.414)
PI 属性	返回圆周率(约等于 3.14159)
SQRT2 属性	返回 2 的平方根(约等于 1.414)
abs(x)方法	返回数的绝对值
acos(x)方法	返回数的反余弦值
exp(x)方法	返回 e 的指数
floor(x)方法	对数进行下舍入
pow(x,y)方法	返回 x 的 y 次幂
random()方法	返回 0～1 之间的随机数
round(x)方法	把数四舍五入为最接近的整数
sqrt(x)方法	返回数的平方根

【例 5-9】　求 2 的 3 次方。

```
<script type="text/javascript">
    document.write(Math.pow(2,3));
</script>
```

7. RegExp 对象

RegExp 对象表示正则表达式，它是对字符串执行模式匹配的强大工具。

创建 RegExp 对象的语法：

```
new RegExp(pattern, attributes);
```

参数 pattern 是一个字符串，指定了正则表达式的模式或其他正则表达式。参数 attributes 是一个可选的字符串，包含属性 "g"、"i" 和 "m"，分别用于指定全局匹配、区分大小写匹配和多行匹配。RegExp 对象常用的属性及方法如表 5-7 所示。

表 5-7　RegExp 对象常用的属性及方法

属性/方法	描　　述
global 属性	RegExp 对象是否具有标志 g
ignoreCase 属性	RegExp 对象是否具有标志 i
lastIndex 属性	一个整数，标示开始下一次匹配的字符位置
multiline 属性	RegExp 对象是否具有标志 m
source 属性	正则表达式的源文本
compile()方法	编译正则表达式
exec()方法	检索字符串中指定的值。返回找到的值，并确定其位置
test()方法	检索字符串中指定的值。返回 true 或 false

【例 5-10】 在字符串中全局搜索"man",并用"person"替换。然后通过 compile()方法,改变正则表达式,用"person"替换"man"或"woman"。

```
<script type="text/javascript">
    var str="Every man in the world! Every woman on earth!";
    patt=/man/g;
    str2=str.replace(patt,"person");
    document.write(str2+"<br />");
    patt=/(wo)?man/g;
    patt.compile(patt);
    str2=str.replace(patt,"person");
    document.write(str2);
</script>
```

5.3 JavaScript 框架

随着 Web 2.0 的快速发展,以及浏览器端所承载的工作越来越大,在不太影响性能的情况下,开发者都习惯尽量让浏览器多完成一些操作,以减轻服务器的压力和带宽费用等。所以 JavaScript 已经成为 Web 开发最基本的要求之一。而在现实的敏捷开发中,我们通常会选择一个 JavaScript 框架来取代烦琐的本地 JavaScript 的编写。这样会节省很多的时间,写的代码也很清晰明了。在对框架深入学习的同时,对 JavaScript 脚本也会理解得更透彻一些。

在 JavaScript 框架出现之前,开发人员只能在页面上进行 jQuery 操作。这种方式很容易引起开发人员对编码操作的困惑,而且不易管理。Backbone 是最初的领跑者,提供了一个基本的结构和组织以及开发者友好的框架,如 Angular 和 Ember,如今得到了许多项目开发者的青睐。我们常常会忽视开发人员的专业背景,而这些专业背景很可能会使他们发现一种流行的框架,这种框架给人以很直观的感觉,并且使得 JavaScript 框架更容易被理解。

那么什么是 JavaScript 框架,JavaScript 框架通常是指以 JavaScript 语言为基础搭建的编程框架。它是预先写好的 JavaScript 代码集。这些 JavaScript 框架也被称为 JavaScript 库,JavaScript 框架帮助开发者快速设计和开发动态网站,给开发者提供很多的便利。JavaScript 库封装了很多预定义的对象和实用函数,能帮助使用者轻松建立有高难度交互的 Web 2.0 特性的富客户端页面,并且兼容几个主要的浏览器。下面是对目前流行的 JavaScript 库的介绍和对比,重点介绍 3 种。

5.3.1 Prototype

Prototype 是最早成型的 JavaScript 库之一,对 JavaScript 的内置对象(例如 String 对象、Array 对象等)做了大量的扩展。现在还有很多项目使用 Prototype。Prototype 可以看作把很多好的、有用的 JavaScript 的方法组合在一起而形成的 JavaScript 库。使用者可以在需要的时候随时将其中的几段代码抽出来放进自己的脚本里。但是由于 Prototype 成型年代早,从整体上对于面向对象的编程思想把握得不是很到位,导致了其结构的松散。不过现在 Prototype 也在慢慢改进。

Prototype 的特点。

（1）对字符串进行各种处理。

（2）使用枚举的方式访问集合对象。

（3）以更简单的方式进行常见的 DOM 操作。

（4）使用 CSS 选择符定位页面元素。

（5）发起 ajax 方式的 HTTP 请求并对响应进行处理。

（6）监听 DOM 事件并对事件进行处理。

5.3.2　YUI

YUI 是由 Yahoo 公司开发的一套完备的、扩展性良好的有高交互性的网页程序工具集。YUI 封装了一系列比较丰富的功能，例如 DOM 操作和 ajax 应用等，同时还包括了几个核心的 CSS 文件。该库本身的文档极其完备，代码编写也非常规范。YUI 包含完整的说明文件，它包含了工具与控件两种元件，以及一些 CSS 资源。

1. YUI 工具

（1）动画：协助实现位置移动、大小改变、透明度和其他的网页效果。

（2）浏览器历史记录管理工具：协助网页程式使用浏览器的上一页与书签工具。

（3）连线工具：协助管理跨浏览器的 XMLHttpRequest 功能。它也整合了表单传送、错误处理、callback 和档案上传。

（4）资料源：给其他组件提供通用可配置接口与种种资料，如从简单的 JavaScript 阵列到线上服务器，并通过 XHR 来互动。

（5）元素：为 DOM 里的 HTML 元素提供包装样式，从而简化一般工作如加入监听者（listener）、对 DOM 操作，以及存取属性。

（6）DOM：为一般的 DOM 脚本作业提供帮助，它包括元素定位与 CSS 样式管理。

（7）即拖即放：为即拖即放的开发（建立与管理可在网页上拖放的物件）提供帮助。

（8）事件：提供开发者对浏览器事件（如鼠标单击与键盘按键）的简易、安全的存取。它也提供相关功能以应付用户出版与订阅自订事件的需求。

2. YUI 控件

（1）自动完成：为用户输入的文字互动提供自动完成功能。它支持广泛的资料源格式。它也透过 XMLHttpReqeust 支持服务器端资料源。

（2）按钮：让用户制作方面的功能像传统 HTML 表单按钮般多样、图形化。

（3）日历：图形式、动态的控制，用于日期选择。

（4）容器：支持大量的 DHTML 视窗规范，包括提示框（Tooltip）、面板、对话框、简易对话框、模组与覆盖层（Overlay）。

（5）资料表：简单且强大的应用程序接口，用来显示网页上屏幕阅读器可存取的表资料。值得关注的功能包括可排序的栏、分页、卷轴、行选取、可放大缩小的栏以及线上编辑。

（6）纪录器：提供一种快速简单的方式来写入日志信息到 Mozilla Firefox 的 Firebug 扩充插件画面终端或者 Safari JavaScript 终端。

（7）表单：提供鼠标光标移过时弹出列表的方式。

（8）滑块：提供一般性滑块组件让用户可在有限范围内以单轴或者双轴选择值。

(9) 分页检视：提供以分页方式来检视内容的功能。

(10) 树状检视：产生目录树，其下面的节点可以缩放。

3. YUI CSS 资源

(1) CSS 页面网格：七种基本线框外带附加组件，支持超过 1000 种的网页布局。

(2) 标准 CSS 字型集：标准化跨浏览器字型家族与尺寸设定。

(3) 标准 CSS 重设：用于移除页边空白并标准化跨浏览器时显示一般元素的问题。

4. YUI 实例

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script src="yui-min.js"></script>
<script>
    YUI().use('node', function (Y) {
        var helloWorld=function(e) {
            e.target.setHTML("修改成功!");
        };
        var myDiv=Y.one("#container");
        myDiv.on("click", helloWorld);
    });
</script>
</head>
<body>
    <div id="container" style="width: 300px; height: 100px; background-
        color:#CCC; padding-top:10px; padding-left:10px">
        单击修改内容
    </div>
</body>
</html>
```

执行以上代码，如图 5-4 所示。

单击之后，结果如图 5-5 所示。



图 5-4　修改网页的内容



图 5-5　内容修改成功

5.3.3　ExtJS

ExtJS 通常简称为 Ext，原本是对 YUI 的扩展，主要用于创建前端用户界面，如今已经发展到可以利用包括 jQuery 在内的各种 JavaScript 框架作为基础库，而 Ext 作为界面的扩展库来使用。Ext 可以用来开发有漂亮外观的客户端应用，能使 B/S 应用更加具有灵活性。但是由于 Ext 侧重于界面，本身比较臃肿，所以使用之前请先权衡利弊。另外，需要注意的

是，Ext 并非完全免费，如果用于商业用途，需要付费并获得授权许可。

1. ExtJS 的特点

（1）纯 HTML/CSS+JavaScript 技术，重新定义表示层的耦合。

（2）基于纯 HTML/CSS+JavaScript 技术，提供丰富的跨浏览器用户界面组件，灵活采用 JSON/XML 数据源进行开发，使得服务端表示层的负荷真正减轻，从而达到客户端的 MVC 应用。

（3）集成多种 JavaScript 底层库，满足开发者不同的需求。

（4）Ext 初期仅是对 YUI 的对话框进行扩展，后来逐渐有了自己的特色，深受网友的喜爱。发展至今，Ext 除 YUI 外还支持 jQuery Prototype 等的 JavaScript 库，让用户可以自由地选择。

（5）多浏览器支持、支持多平台下的主流浏览器。

2. ExtJS 的优点

（1）UI 组件丰富，外观漂亮。ExtJS 库有着丰富且漂亮的 UI 组件，大大缩短了开发周期，而且组件拥有漂亮的布局，经过简单的调用与配置就可以实现不错的界面布局。ExtJS 提供的各种组件可以用更加标准的方式展示数据，降低了开发难度。

（2）浏览器兼容性好。使用 ExtJS 对浏览器没有任何要求。可以说是一种绿色的客户端实现方式。ExtJS 基本可以运行于现在主流的浏览器中。

（3）有很多动画。效果做得很不错，提高了用户的感知度。

（4）与后台代码无关。不管后台用什么语言开发都不会受影响。

（5）将 Web 程序向桌面系统转化。ExtJS 最大的优势在于它将 Web 应用程序的操作方式向传统桌面应用程序的操作方式进行转化甚至消除了这种差异，从根本上提高了用户的使用体验，这是 ExtJS 应用前景广阔的主要原因。

（6）相对丰富的文档和示例。毫无疑问，ExtJS 附带的例子和开发文档十分吸引程序员，它的文档做得确实不错。

3. ExtJS 的缺点

（1）体积较大，速度稍慢。由于使用了大量的 UI 组件，所以体积较大，导致页面加载速度比较慢。

（2）收费。因为 ExtJS 太优秀了，所以从 ExtJS 2.0 以后的版本都是收费的。也许这不能算是它的缺点，但这确实阻碍了它的推广与应用。

（3）没有合适的开发利器。毫无疑问，一个好的开发工具可以大大地提高编码的速度，但是对于 ExtJS，始终没有一个完美的开发工具，可以推荐的有 Aptana Studio、Spket IDE 和 Spket 提供的提示文件，但是都各有优缺点，都不完美，只能一边看 SDK 一边写代码。

（4）没有界面设计工具。虽然有人提供了一个在线的界面设计工具，但是和 Visual Studio 提供的 ASP.NET 设计工具相比，可以说有天壤之别。因此，只能一边预览，一边写代码。

（5）文档不全。虽然 ExtJS 提供的文档很丰富，但还是跟不上源代码的更新速度，所以，经常要通过看源代码，调试才能真正解决问题。

（6）不能编译。这一点可以说是 JavaScript 的缺点（如果能编译，就不叫 JavaScript 了）。在实际的开发过程中，经常会输错一些代码，比如有大小写错误等，不能通过编译得到

反馈，只能在运行时排错，导致开发的效率比较低下。

4. ExtJS 实例

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
<script src="ext-all.js" type="text/jscript"></script>
<script src="ext-lang-zh_CN.js" type="text/javascript"></script>
<link href="ext-all.css" rel="stylesheet" type="text/css" />
<script type="text/javascript">
    Ext.onReady(function()
    {
        Ext.MessageBox.alert('helloworld', 'Hello World.');
    });
</script>
</head>
<body>
</body>
</html>
```

执行以上代码，创建一个窗口，效果如图 5-6 所示。



图 5-6　ExtJS 窗口

5.3.4　jQuery

jQuery 是一个快速、简洁的 JavaScript 框架，是继 Prototype 之后又一个优秀的 JavaScript 代码库（或 JavaScript 框架）。jQuery 设计的宗旨是倡导写更少的代码，做更多的事情。它封装了 JavaScript 常用的功能代码，提供一种简便的 JavaScript 设计模式，优化了 HTML 文档操作、事件处理、动画设计和 ajax 交互，是一个轻量级框架，兼容 CSS 3，兼容各种浏览器。但是不能向后兼容，新版本不能兼容前期的版本，在同一页面上使用多个插件时，很容易碰到冲突现象；在大型框架中，jQuery 核心代码库对动画和特效的支持相对较差。

1. jQuery 的特点

（1）一款轻量级的 JavaScript 框架。jQuery 核心 JavaScript 文件才几十 KB，不会影响页面加载的速度。与 ExtJS 相比要轻便很多。

（2）丰富的 DOM 选择器。jQuery 的选择器用起来很方便，好比要找到某个 DOM 对象的相邻元素 JavaScript 可能要写好几行代码，而 jQuery 一行代码就可以解决问题。再比如要将一个表格的隔行变色，jQuery 也是一行代码就可以解决问题。

（3）链式表达式。jQuery 的链式操作可以把多个操作写在一行代码里，会更加简洁。

（4）事件、样式、动画支持。jQuery 还简化了 JavaScript 操作 CSS 的代码，并且代码的可读性也比 JavaScript 要强。

（5）支持 ajax 操作。jQuery 简化了 ajax 操作，后台只需返回一个 JSON 格式的字符串

就能完成与前台的通信。

(6) 跨浏览器兼容。jQuery 基本兼容了现在主流的浏览器,不用再为浏览器的兼容问题而伤透脑筋。

(7) 插件扩展开发。jQuery 有着丰富的第三方的插件,例如:树形菜单、日期控件、图片切换插件、弹出窗口等基本前台页面上的组件都有对应的插件,用 jQuery 插件做出来的效果很炫,并且可以根据自己的需要去改写和封装插件,简单、实用。

(8) 可扩展性强。jQuery 提供了扩展接口:jQuery.extend(object),可以在 jQuery 的命名空间上增加新函数。jQuery 的所有插件都是基于这个扩展接口开发的。

2. jQuery 的优点

(1) jQuery 实现了脚本与页面的分离。这样我们可以实现灵活性非常强的清晰页面代码。jQuery 让 JavaScript 代码从 HTML 页面代码中分离出来,就像数年前 CSS 让样式代码与页面代码分离开一样。

(2) 最少的代码做最多的事情。最少的代码做最多的事情,这是 jQuery 的口号,而且名副其实。使用它的高级选择器,开发者只需编写几行代码就能实现令人惊奇的效果。开发者无须过于担忧浏览器的差异,它除了仍完全支持 ajax,而且拥有许多提高开发者编程效率的其他抽象概念。

(3) 性能支持得比较好。在大型 JavaScript 框架中,jQuery 对性能的理解最好。尽管不同版本拥有众多新功能,其最精简版本只有 18KB 大小,这个数字已经很难再减少。jQuery 的每一个版本都有极大的性能提高。如果将其与新一代具有更快 JavaScript 引擎的浏览器(如火狐 3 和谷歌 Chrome)配合使用,开发者在创建 Web 应用时将拥有明显的速度优势。

(4) 它是一个“标准”。之所以使用引号,是因为 jQuery 并非只有一个官方标准。但是业内对 jQuery 的支持已经非常广泛。谷歌不但自己使用它,还提供给用户使用。另外戴尔、新闻聚合网站 Digg、WordPress、Mozilla 和许多其他厂商也在使用它。微软甚至将它整合到 Visual Studio 中。如此多的重量级厂商支持该框架,用户完全可以对其的未来放心,大胆地将其投入使用。

(5) 插件开发。基于 jQuery 开发的插件目前已经大约有数千个。开发者可使用插件来完成表单确认、图表种类、字段提示、动画、进度条等任务。jQuery 社区已经成长为一个生态系统。这一点进一步证明了它是一个安全的选择。而且 jQuery 正在主动与“竞争对手”合作,例如 Prototype。它们似乎在推进 JavaScript 的整体发展,而不仅仅是在图谋一己之私。

(6) 节约学习成本。当然要想真正学习 jQuery,开发者还是需要投入一点时间,如果要编写大量代码或自主插件,更是如此。但是,开发者可以采取“各个击破”的方式,而且 jQuery 提供了大量的示例代码,入门是一件非常容易的事情。建议开发者在编写某类代码前,首先看一下是否有类似插件,然后看一下实际的插件代码,了解一下其工作原理。简而言之,学习 jQuery 不需要开发者投入太多,就能够迅速开始开发工作,然后逐渐提高技巧。

(7) 让 JavaScript 编程变得有趣。使用 jQuery 是一件充满乐趣的事情。jQuery 简洁而强大,开发者能够迅速得到自己想要的结果。它解决了许多 JavaScript 难题。通过一些基础性的改进,开发者可以真正去思考开发下一代 Web 应用,不再因为语言或工具的差劲

而烦恼。应相信它的“用最少的代码做最多的事情”的口号。

3. jQuery 的缺点

(1) 不能向后兼容。每一个新版本不能兼容早期的版本。举例来说，有些新版本不再支持某些选择器，新版 jQuery 却没有保留对它们的支持，而只是简单地将其移除。这可能会影响到开发者已经编写好的代码或插件。

(2) 插件兼容性。与上一点类似，当新版 jQuery 推出后，如果开发者想升级，要看插件作者是否支持。通常情况下，在最新版的 jQuery 下，现有插件可能无法正常使用。开发者使用的插件越多，这种情况发生的概率也越高。

(3) 多个插件冲突。在同一页面上使用多个插件时，很容易碰到冲突现象，尤其是这些插件依赖相同的事件或选择器时最为明显。这虽然不是 jQuery 自身的问题，但又确实是一个难于调试和解决的问题。

(4) 对动画和特效的支持差。在大型框架中，jQuery 核心代码库对动画和特效的支持相对较差。但实际上这不是根本性的问题。目前有一个单独的 jQuery UI 项目和众多插件来弥补这一点。

4. jQuery 实例

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script type="text/javascript" src="/jquery/jquery.js"></script>
<script type="text/javascript">
    $(document).ready(function(){
        $("button").click(function(){
            $(".test").hide();
        });
    });
</script>
</head>
<body>
    <h2 class="test">这是一个标题</h2>
    <p class="test">这是一个段落</p>
    <p>这是另一个段落</p>
    <button type="button">单击隐藏段落</button>
</body>
</html>
```

执行代码，效果如图 5-7 所示。

单击按钮之后，结果如图 5-8 所示。

图 5-7　jQuery 实例

图 5-8　隐藏段落

5.3.5　Dojo

Dojo 的强大之处在于 Dojo 提供了很多其他 JavaScript 库所没有提供的功能。例如离线存储的 API、生成图标的组件、基于 SVG/VML 的矢量图形库和对 Comet 的支持等。Dojo 是一款非常适合企业级应用的 JavaScript 库，并且得到了 IBM、SUN 和 BEA 等一些大公司的支持。但是 Dojo 的缺点也是很明显的：学习曲线陡，文档不齐全，最严重的就是 API 不稳定，每次升级都可能导致已有的程序失效。但是自从 Dojo 的 1.0.0 版出现以后，情况有所好转。Dojo 还是一个很有发挥潜力的库。

1. Dojo 的特点

（1）Dojo 是一个符合 AMD 规范的企业级框架（Dojo 是一个重量级框架）。

（2）Dojo 全面支持异步加载 JS 机制（即支持通过 require 异步加载 JS 模块，通过 define 定义符合 AMD 规范的标准 JS 直接对象（模块））。

（3）Dojo 不仅提供了针对 JavaScript 的语句优化（Dojo 核心语法与 jQuery 完全不同，需要额外学习），还提供了所有 UI 组件。

（4）Dojo 支持 IE 6 以上版本浏览器，Dojo 帮助我们处理浏览器兼容性问题，所以不需要担心 Web 页面是否在某些浏览器中可用。

（5）Dojo 提供了打包工具，可以优化 JavaScript 代码，并且只生成部署应用程序所需的最小 Dojo 包集合。

2. Dojo 与 jQuery 相比所具有的优点

（1）Dojo 支持 IE 6 版本以上的浏览器，jQuery 不再支持 IE 9 以下版本的浏览器（国内 IE 9 以下版本的浏览器份额依旧占据半壁江山）。

（2）Dojo 不仅提供 jQuery 的 JS 优化操作，还提供 UI 组件。

（3）Dojo 中的所有功能都基于异步 JS 实现，jQuery 需要借助第三方的异步加载框架实现异步加载 JS。

5.3.6　MooTools

MooTools 是一套轻量、简洁、模块化和面向对象的 JavaScript 框架。MooTools 的语法几乎和 Prototype 一样，但提供了更为强大的功能、更好的扩展性和兼容性。其模块化思想非常优秀，核心代码只有 8KB 大小。无论用到哪个模块都可及时导入，即使是完整版大小也不超过 160KB。MooTools 具有完全彻底的面向对象的编程思想，语法简洁直观，文档完善，是一个非常不错的 JavaScript 库。

总之，每个 JavaScript 库都有各自的优点和缺点，同时也有各自的支持者和反对者。

5.4　综合实例

本节我们通过一个实例来体会 JavaScript 框架编程的优越性。

5.4.1 需求描述

利用 JavaScript 的框架 ExtJS 完成表单的制作，表单样式如图 5-9 所示。验证文本框为必输入内容，验证通过后，单击“保存”按钮，弹出提示框。

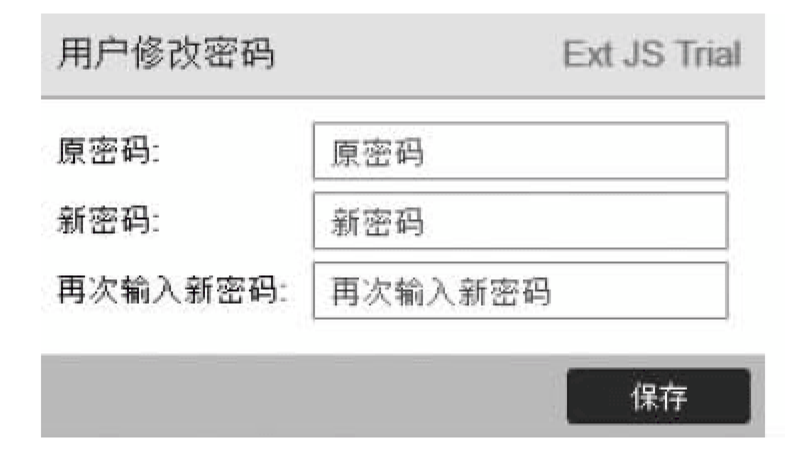


图 5-9 ExtJS 表单窗口

5.4.2 分析及实现

(1) 从官网下载 ExtJS 包 extjs5，这里采用的是 5.0 版本。

下载地址：http://extjs.org.cn/

(2) 解压 ExtJS 包，找到 ext-all.js 基础包(\ext-5.0.0\build)和 ext-all-debug.js 基础包。后面的基础包开发的时候使用，报错会详细些(\ext-5.0.0\build)。再选一个合适的主题，这里使用了 crisp，找到 ext-theme-crisp-all.css 和 images 文件(\packages\ext-theme-crisp\build\resources)。

(3) 新建 index.html 页面并引用 ext-all-debug.js、ext-theme-crisp-all.css。新建 index.js 并应用启动设置文件。新建 app 文件夹，用于放置 controller 和 view 文件夹。

index.html 的内容如下：

```
<!DOCTYPE HTML>
<html>
<head>
    <title>demo</title>
    <link href="../Ext/ext-theme-crisp-all.css" rel="stylesheet" type=
    "text/css" />
    <script src="../Ext/ext-all-debug.js" type="text/javascript">
    </script>
    <scriptsrc="index.js" type="text/javascript"></script>
</head>
<body>
</body>
</html>
```

在 index.html 中引入了 ExtJS 的函数库以及 CSS 样式表，引入了 index.js 文件。

index.js 的内容如下：

```
Ext.Loader.setConfig({
    enabled: true
});

Ext.application({
    name: 'Demo1',
    appFolder: 'app',
    models: [
    ],
```

```
    stores: [
    ],
    controllers: [
        'MyController'
    ],
    views: [
        'MyViewport'
    ],
    launch: function () {
        var app=new Demo1.view.MyViewport();
    }
});
```

(4) 创建视图、控制器。在 view 文件夹下添加视图 MyViewport.js,以下代码在视图里面加了一个简单的表单。

```
Ext.define('Demo1.view.MyViewport', {
    extend: 'Ext.container.Viewport',
    initComponent: function () {
        var me=this;
        Ext.applyIf(me, {
            items: [
                {
                    xtype: 'form',
                    title: '用户修改密码',
                    width: 300,
                    bodyPadding: 10,
                    defaultType: 'textfield',
                    border: false,
                    items: [
                        {
                            allowBlank: false,
                            id: 'txtPwdOld',
                            fieldLabel: '原密码',
                            name: 'pwdOld',
                            labelWidth: 100,
                            emptyText: '原密码',
                            inputType: 'password'
                        },
                        {
                            allowBlank: false,
                            id: 'txtPwdNew',
                            fieldLabel: '新密码',
                            name: 'pwdNew',
                            labelWidth: 100,
                            emptyText: '新密码',
                            inputType: 'password'
                        },
```

```
                        {
                            allowBlank: false,
                            id: 'txtPwdNew2',
                            fieldLabel: '再次输入新密码',
                            name: 'pwdNew2',
                            labelWidth: 100,
                            emptyText: '再次输入新密码',
                            inputType: 'password'
                        }
                    ],
                    buttons: [
                        {
                            text: '保存',
                            id:'btnSavePwd'
                        }
                    ]
                }
            ]
        });

        this.callParent(arguments);
    }

});
```

在 controller 文件夹下添加控制器 MyController. js。程序代码都可以写在控制器里面，用得最多的就是监听控件事件。对表单中的“保存”按钮要监听单击事件。

```
Ext.define('Demo1.controller.MyController', {
    extend: 'Ext.app.Controller',

    init: function (application) {
        this.control({
            '[id=btnSavePwd]': {
                click: this.onOK
            }
        });
    },
    //保存
    onOK: function (obj) {
        var form=obj.up('form').getForm();
        if (form.isValid()) {
            Ext.Msg.alert('信息提示', '已保存');
        }
    }
});
```

到这里程序已经可以运行了。运行 index. html，出现如图 5-10 所示的窗口。

如果文本框中没有输入数据，会显示提示信息，如图 5-11 所示。



图 5-10　index 页面的内容



图 5-11　验证窗口

所有文本框都输入数据后，单击“保存”按钮，弹出确认窗口，如图 5-12 所示。



图 5-12　确认窗口

本章小结

本章主要介绍了 JavaScript 的面向对象编程，JavaScript 是面向对象的语言，但是不使用类，可直接定义对象，并使用对象的方法和属性，另外，JavaScript 也有丰富的内置对象，包括数组、字符串、日期等对象，这些对象提供了丰富的属性和方法，能快速地实现相应的需求。本章还介绍了常用的 JavaScript 框架，包括 Prototype、YUI、ExtJS、jQuery、Dojo 及 MooTools，最后用了一种框架演示了框架的使用方法。

第 6 章　jQuery 编程

6.1　jQuery 简介

随着 Web 2.0 的兴起，JavaScript 脚本语言越来越受到 Web 前端开发者的青睐。各种 JavaScript 框架也蓬勃发展起来。从早期的面向对象框架 Prototype、Dojo 到 2006 年的 jQuery，再到 2007 年的 ExtJS，互联网正在掀起一场 JavaScript 风暴。jQuery 以其“写得少，做得多”的独特风格，受到越来越多前端开发爱好者的追捧。

jQuery 由 John Resig、Brandon Aaron 和 Jorn Zaefferer 创建，它是 JavaScript 的一个类库，是继 Prototype 之后又一个优秀的 JavaScript 库，是一个由 John Resig 创建于 2006 年 1 月的开源项目。现在的 jQuery 团队主要包括核心库、UI 和插件等开发人员以及推广和网站设计维护人员。jQuery 凭借简洁的语法和跨平台的兼容性，极大地简化了 JavaScript 开发人员遍历 HTML 文档、操作 DOM、处理事件、执行动画和开发 ajax 的操作。其独特而又优雅的代码风格改变了 JavaScript 程序员的设计思路和编写程序的方式。总之，无论是网页设计师、后台开发者、业余爱好者还是项目管理者，也无论是 JavaScript 初学者还是 JavaScript 高手，都有足够的理由去学习 jQuery。其优势体现为：

jQuery 强调的理念是“写得少、做得多”。jQuery 独特的选择器、链式的 DOM 操作、事件处理机制和封装完善的 ajax 都是其他 JavaScript 库望尘莫及的。

6.2　jQuery 的基本功能

6.2.1　引用 jQuery 类库

jQuery 提供了很多 JavaScript 类库，这些类库中包含大量的应用程序编程接口(application programming interface，API)。要使用类库中的 API，不需要像传统软件一样安装，只需把下载的 jQuery.js 放到网站上的一个公共的位置。当要在某个页面上使用 jQuery 时，只需要在相关的 HTML 文档中引入该库文件即可。

在实际项目中，读者可以根据实际需要调整 jQuery 的库路径。如 jQuery.html 文件中需要引用 jQuery 库文件，就在该文档的<head></head>标签中添加下面的代码即可。

```
<head>
    <script src="Scripts/jQuery.min.js" type="text/javascript"></script>
</head>
```

jQuery 库文件版本自发布以来，已更新到当前的 3.0 版本，jQuery 2.0 以上版本不再支持 IE 6/7/8，本书的案例如果没有特殊说明，则默认采用 jQuery 1.10.2 版本，引用文件时使用该版本的压缩版文件 jQuery.min.js。

6.2.2　第一个 jQuery 程序

在 jQuery 程序中使用 $ 符号代替 jQuery，这是 jQuery 的一个简写形式，例如 $("#head")和 jQuery("#head")是等价的，如果没有特殊说明，程序中的 $ 符号都是 jQuery 的一个简写形式。

第一个 jQuery 程序的代码如例 6-1 所示。

【例 6-1】　第一个 jQuery 程序。

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>第一个 jQuery 程序</title>
    <script type="text/javascript" src="Scripts/jQuery.min.js"></script>
</head>
<body>
<script language="javascript" type="application/javascript">
    $(document).ready(function() {
        alert("Hello jQuery !");
    });
</script>
</body>
</html>
```

程序的运行结果如图 6-1 所示。



图 6-1　程序的输出结果

以上程序中的如下代码。

```
$(document).ready(function(){
    //...
});
```

这段代码的作用类似于传统 JavaScript 中的 window.onload 事件，不过与 window.onload 还是有些区别。表 6-1 对它们进行了简单对比。

表 6-1　window. onload 与 $ (document). ready()的对比

对比选项	window. onload	$ (document). ready();
执行时机	必须等待网页中所有内容加载完毕后(包含图片)才能执行	网页中所有 DOM 结构绘制完毕后就执行,DOM 元素关联的东西可能并没有加载完

编写个数	不能同时编写多个。以下代码无法正确执行:
	window. onload=function(){
	　　alert("Hello")
	};
	window. onload=function(){
	　　alert("jQuery")
	};
	执行以上程序,结果只会输出 jQuery
		能同时编写多个。以下代码可以正确执行:
		$ (document). ready(function(){
		　　alert("Hello World!");
		});
		$ (document). ready(function(){
		　　alert("HellojQuery!");
		});
		执行以上程序,结果两次都有输出

简化写法	无	$ (document). ready(function(){
		　　…
		})
		可以简写成:
		$ (function(){
		…
		})

6.2.3　jQuery 选择器

当要操作 HTML 页面上的元素时,需快速定位到页面上的目标元素,在 JavaScript 中一般使用 document. getElementById 获得这些元素。但要灵活获取 HTML 页面中的元素,使用 JavaScript 就显得很麻烦,如果使用 jQuery 选择器,操作过程就会更简单。

jQuery 中的选择器完全继承了 CSS 样式的风格。利用 jQuery 选择器,可以非常便捷和快速地找出特定的 DOM 元素,然后为它们添加相应的行为,而无须担心浏览器是否支持这一选择器。学会使用选择器是学习 jQuery 的基础,jQuery 的行为规则都必须在获取到元素后才能生效。在 jQuery 中,有多种方法可以快速获取元素,元素选择器和属性选择器允许开发者通过标签名、属性名或内容对 HTML 元素进行选择,选择后可对 HTML 元素组或单个元素进行操作。

下面来看一个简单的例子,代码如例 6-2 所示。

【例 6-2】　JavaScript 执行事件。

```
<script type="text/javascript">
    function test(){
        alert("Hello JavaScript");
    }
</script>
<p onclick="test();">单击我</p>
```

本段代码的作用是为<p>元素设置一个 onClick 事件,当单击此元素时,会弹出一个对话框。像上面这样把 JavaScript 代码和 HTML 代码混杂在一起的做法同样也不妥,因为

这样并没有将网页内容和行为分离。如果使用如下的代码，效果就明显不一样，并实现了 JavaScript 代码和 HTML 代码的分离。

【例 6-3】 jQuery 执行事件。

```
<p class="testClass">jQuery Test</p>
<script type="text/javascript">
    //给 class 为 testClass 的元素添加行为
    $(".testClass").click(function(){
        alert("Hello jQuery!");
    })
</script>
```

下面将 CSS 的写法和 jQuery 的写法进行比较。

CSS 获取到元素的代码如下：

```
.testClass {
    ...//给 testClass 类中的元素添加样式
}
```

jQuery 获取到元素的代码如下：

```
$(".testClass"){
    ...//给 testClass 类中的元素添加行为
}
```

jQuery 选择器的写法与 CSS 选择器的写法十分类似，只不过两者的作用不同，CSS 选择器找到元素后是添加样式，而 jQuery 选择器找到元素后是添加行为。需要特别说明的是，jQuery 中涉及操作 CSS 样式部分比单纯的 CSS 功能更为强大，并且拥有跨浏览器的兼容性。

jQuery 选择器分为基本选择器、层次选择器、过滤选择器和表单选择器。

1．基本选择器

基本选择器是 jQuery 中最常用的选择器，也是最简单的选择器，它通过元素 id、class 和标签名等来查找 DOM 元素。在网页中，每个 id 名称只能使用一次，class 允许重复使用。基本选择器的介绍说明如表 6-2 所示。

表 6-2　基本选择器的介绍说明

选择器	描述	返回	示例
#id	根据给定的 id 匹配一个元素	单个元素	$("#test")用于选取 id 为 test 的元素
.class	根据给定的类名匹配元素	集合元素	$(".test")用于选取所有 class 为 test 的元素
element	根据给定的元素名匹配元素	集合元素	$("p")用于选取所有的<p>元素

续表

选　择　器	描　　述	返　回	示　　例
*	匹配所有元素	集合元素	$("*")用于选取所有的元素
selector1,selector2,…	将每一个选择器匹配到的元素合并后再一起返回	集合元素	$("div,span,p.myClass")用于选取所有＜div＞、＜span＞和拥有 class 为 myClass 的＜p＞标签的一组元素

【例 6-4】 id 选择器的使用。

以下代码将 id="one"的元素背景色设置为黑色。

```
<div id="one" style="color:#fff;">修改我的背景色</div>
<div id="two">不修改我的背景色</div>
<script type="text/javascript">
    $(document).ready(function () {
        $('#one').css('background', '#000');
    });
</script>
```

程序的运行效果如图 6-2 所示。

修改我的背景色
不修改我的背景色

图 6-2　id 选择器的使用

2. 层次选择器

层次选择器可以通过 DOM 元素之间的层次关系来获取特定元素,例如后代元素、子元素、相邻元素和兄弟元素等。层次选择器的说明如表 6-3 所示。

表 6-3　层次选择器的说明

选　择　器	描　　述	返　回	示　　例
$("ancestor descendant")	选取 ancestor 元素里的所有 descendant(后代元素)	集合元素	$("div span")用于选取＜div＞里的所有的＜span＞元素
$("parent＞child")	选取 parent 元素下的 child(子)元素	集合元素	$("div＞span")用于选取＜div＞元素下元素名是＜span＞的子元素
$("prev+next")	选取紧接在 prev 元素后的 next 元素	集合元素	$(".one+div")用于选取 class 为 one 的下一个＜div＞元素
$("prev～siblings")	选取 prev 元素之后的所有 siblings 元素	集合元素	$("#two～div")用于选取 id 为 two 的元素后面的所有＜div＞兄弟元素

【例 6-5】 层次选择器的使用。

```
<div style="border:1px solid #000;">
    <span>123</span>
```

```
    <p>
        <span>456</span>
    </p>
</div>
<script type="text/javascript">
    $(document).ready(function () {
        //选取 div 下的第一代 span 元素,并将字体大小改为 16px
        $('div>span').css('color', '#FF0000');
    });
</script>
```

以上面代码中,只有第一个 span 的字体会变成 16px,第二个 span 不属于 div 的一代子元素,字体大小保持不变。程序的运行效果如图 6-3 所示。

123

456

图 6-3　层次选择器的使用

3. 过滤选择器

过滤选择器主要是通过特定的过滤规则来筛选出所需的 DOM 元素。过滤规则与 CSS 的伪类选择器语法相同,即选择器都以一个冒号(:)开头。按照不同的过滤规则,过滤选择器可以分为基本过滤、内容过滤、可见性过滤、属性过滤、子元素过滤和表单对象属性过滤选择器。

(1) 基本过滤选择器。基本过滤选择器的使用频率很高,包括的内容说明详见表 6-4 所示。

表 6-4　基本过滤选择器的说明

选择器	描述	返回	示例
:first	选取第一个元素	单个元素	$("div:first")用于选取所有<div>元素中的第一个<div>元素
:last	选取最后一个元素	单个元素	$("div:last")用于选取所有<div>元素中的最后一个<div>元素
:not(selector)	去除所有与给定选择器匹配的元素	集合元素	$("input:not(.myClass)")用于选取 class 不是 myClass 的<input>元素
:even	选取索引是偶数的所有元素,索引从 0 开始	集合元素	$("input:even")用于选取索引是偶数的<input>元素
:odd	选取索引是奇数的所有元素,索引从 0 开始	集合元素	$("input:odd")用于选取索引是奇数的<input>元素
:eq(index)	选取索引等于 index 的元素(index 从 0 开始)	单个元素	$("input:eq(1)")用于选取索引等于 1 的<input>元素
:gt(index)	选取索引大于 index 的元素(index 从 0 开始)	集合元素	$("input:gt(1)")用于选取索引大于 1 的<input>元素(注:大于 1 并不包括 1)
:lt(index)	选取索引小于 index 的元素(index 从 0 开始)	集合元素	$("input:lt(1)")用于选取索引小于 1 的<input>元素(注:小于 1 并不包括 1)
:header	选取所有的标题元素,例如 h1、h2、h3 等	集合元素	$(":header")用于选取网页中所有的<h1>,<h2>,<h3>,…
:animated	选取当前正在执行动画的所有元素	集合元素	$("div:animated")用于选取正在执行动画的<div>元素

（2）内容过滤选择器。内容过滤选择器的过滤规则主要体现在它所包含的子元素或文本内容上。内容过滤选择器的说明如表 6-5 所示。

表 6-5　内容过滤选择器的说明

选择器	描　述	返　回	示　例
:contains(text)	选取含有文本内容为 text 的元素	集合元素	$("div:contains('我')")用于选取含有文本"我"的<div>元素
:empty	选取不包含子元素或者文本的空元素	集合元素	$("div:empty")用于选取不包含子元素(文本元素)的<div>空元素
:has(selector)	选取含有选择器所匹配的元素的元素	集合元素	$("div:has(p)")用于选取含有<p>元素的<div>元素
:parent	选取含有子元素或者文本的元素	集合元素	$("div:parent")用于选取拥有子元素(包含文本元素)的<div>元素

（3）可见性过滤选择器。可见性过滤选择器是根据元素的可见和不可见状态来选择相应的元素。可见性过滤选择器的说明如表 6-6 所示。

表 6-6　可见性过滤选择器的说明

选择器	描　述	返　回	示　例
:hidden	选取所有不可见的元素	集合元素	$(":hidden")用于选取所有不可见的元素。包括<input type="hidden"/>，<div style="display:none;">和<div style="visibility:hidden;">等元素。如果只想选取<input>元素，可以使用$("input:hidden")
:visible	选取所有可见的元素	集合元素	$("div:visible")用于选取所有可见的<div>元素

（4）属性过滤选择器。属性过滤选择器的过滤规则是通过元素的属性来获取相应的元素。属性过滤选择器的说明如表 6-7 所示。

表 6-7　属性过滤选择器的说明

选择器	描　述	返　回	示　例
[attribute]	选取拥有此属性的元素	集合元素	$("div[id]")用于选取拥有属性 id 的元素
[attribute=value]	选取属性的值为 value 的元素	集合元素	$("div[title=test]")用于选取属性 title 为 test 的<div>元素
[attribute!=value]	选取属性的值不等于 value 的元素	集合元素	$("div[title!=test]")用于选取属性 title 不等于 test 的<div>元素(注意：没有属性 title 的<div>元素也会被选取)
[attribute^=value]	选取属性的值以 value 开始的元素	集合元素	$("div[title^=test]")用于选取属性 title 以 test 开始的<div>元素

续表

选　择　器	描　　述	返　回	示　　例
[attribule $ = value]	选取属性的值以 value 结束的元素	集合元素	$ ("div[title $ = test]")用于选取属性 title 以 test 结束的<div>元素
[attribute * = value]	选取属性的值含有 value 的元素	集合元素	$ ("div[title * = test]")用于选取属性 title 含有 test 的<div>元素
[selector1],[selector2],…	用属性选择器合并成一个复合属性选择器,满足多个条件。每选择一次,缩小一次范围	集合元素	$ ("div[id] [title $ = 'test']")用于选取拥有属性 id,并且属性 title 以 test 结束的<div>元素

(5) 子元素过滤选择器。子元素过滤选择器的过滤规则相对于其他的选择器稍微有些复杂,不过没关系,只要将元素的父元素和子元素区分清楚,那么使用起来也非常简单。另外还要注意它与普通的过滤选择器的区别。

子元素过滤选择器的说明如表 6-8 所示。

表 6-8　子元素过滤选择器

选　择　器	描　　述	返　回	示　　例
:nth-child(index/even/odd/equation)	选择每个父元素下的第 index 个子元素或者奇偶元素(index 从 1 算起)	集合元素	":eq(index)"只匹配一个元素,而":nth-child"将为每一个父元素匹配子元素,并且":nth-child(index)"的 index 是从 1 开始的,而":eq(index)"是从 0 算起的
:first-child	选取每个父元素的第 1 个元素	集合元素	":first"只返回单个元素,":first-child"选择符将为每个父元素匹配第 1 个子元素。例如 $ ("ul li:first-child");用于选取每个 中的第 1 个 元素
:last-child	选取每个父元素的最后一个元素	集合元素	同样,":last"只返回单个元素,而":last-child"选择符将为每个父元素匹配最后一个子元素。例如 $ ("ul li:last-child")选择每个 中的最后一个 元素
:only-child	如果某个元素是其父元素的唯一的子元素,那么将会被匹配。如果父元素中含有其他元素,则不会被匹配	集合元素	$ ("ul li:only-child")在 中选取作为唯一子元素的 元素

(6) 表单对象属性过滤选择器。此选择器主要是对所选择的表单元素进行过滤,例如选择被选中的下拉框、多选框等。表单对象属性过滤选择器的说明如表 6-9 所示。

表 6-9　表单对象属性过滤选择器的说明

选择器	描　　述	返　回	示　　例
:enabled	选取所有可用元素	集合元素	$("#form1:enabled")用于选取 id 为 form1 的表单内的所有可用元素
:disabled	选取所有不可用元素	集合元素	$("#form2:disabled")用于选取 id 为 form2 的表单内的所有不可用元素
:checked	选取所有被选中的元素(单选框、复选框)	集合元素	$("input:checked")用于选取所有被选中的＜input＞元素
:selected	选取所有被选中的选项元素(下拉列表)	集合元素	$("select:selected")用于选取所有被选中的选项元素

【例 6-6】　过滤选择器的使用。

```
<table width="200" cellpadding="0" cellspacing="0"
    style="border:1px solid #000;">
    <tbody>
        <tr><td>A</td></tr>
        <tr><td>B</td></tr>
        <tr><td>C</td></tr>
        <tr><td>D</td></tr>
    </tbody>
</table>
<script type="text/javascript">
    $(document).ready(function () {
        //偶数行颜色
        $('tr:even').css('background', '#EEE');
        //奇数行颜色
        $('tr:odd').css('background', '#DADADA');
    });
</script>
```

以上代码可使用户设置表格中偶数和奇数行的背景颜色,索引从 0 开始,even 表示偶数,odd 表示奇数,程序的运行效果如图 6-4 所示。

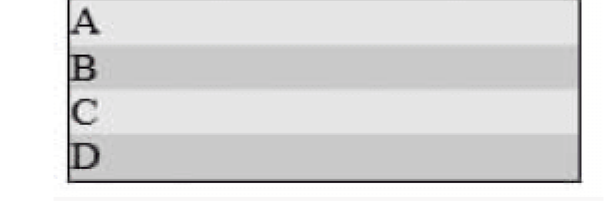


图 6-4　过滤选择器的使用

4. 表单选择器

为了使用户能够更加灵活地操作表单,jQuery 中专门加入了表单选择器,能极其方便地获取到表单的某个或某类型的元素,示例如表 6-10 所示。

表 6-10　表单对象属性过滤示例

选择器	描　　述	返　回	示　　例
:input	选取所有的＜input＞、＜textarea＞、＜select＞和＜button＞元素	集合元素	$(":input")用于选取所有的＜input＞、＜textarea＞、＜select＞和＜button＞元素
:text	选取所有的单行文本框	集合元素	$(":text")用于选取所有的单行文本框

续表

选择器	描　　述	返　回	示　　例
:password	选取所有的密码框	集合元素	$(":password")用于选取所有的密码框
:radio	选取所有的单选框	集合元素	$(":radio")用于选取所有的单选框
:checkbox	选取所有的多选框	集合元素	$(":checkbox")用于选取所有的复选框
:submit	选取所有的“提交”按钮	集合元素	$(":submit")用于选取所有的“提交”按钮
:image	选取所有的图像按钮	集合元素	$(":image")用于选取所有的图像按钮
:reset	选取所有的“重置”按钮	集合元素	$(":reset")用于选取所有的“重置”按钮
:button	选取所有的按钮	集合元素	$(":button")用于选取所有的按钮
:file	选取所有的上传域	集合元素	$(":file")用于选取所有的上传域
:hidden	选取所有的不可见元素	集合元素	$(":hidden")用于选取所有的不可见元素

【例 6-7】　表单选择器的使用。

```
<fieldset style="width: 300px;">
    <legend>账户登录</legend>
    <div>
        <label>用户名:</label><input type="text"/>
    </div>
    <br>
    <div>
        <label>密     码:</label><input type="password"/>
    </div>
</fieldset>
<script type="text/javascript">
   $(document).ready(function() {
        $('input:text').css('border', '3px solid #000');
        $('input:password').css('border', '1px solid #000');
   });
</script>
```

以上代码通过表单选择器的 $('input:text')方法设置用户名文本框的边框线为 3px，使用表单选择器的 $('input:password')方法设置密码文本框的边框线为 1px，程序的运行效果如图 6-5 所示。

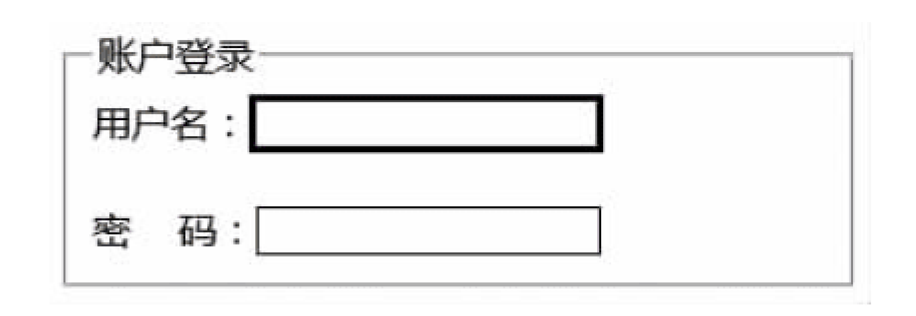


图 6-5　表单选择器的使用

6.2.4　jQuery 事件方法

事件方法会触发匹配元素的事件，或将函数绑定到所有匹配元素的某个事件，jQuery 提供了丰富多样的事件处理方法。

1. 载入事件 ready(fun)

ready(fun)在 DOM 载入就绪并可以查询及操纵时绑定一个要执行的函数，这是事件模块中最重要的一个函数，因为它可以极大地提高 Web 应用程序的响应速度。简单地说，这个方法纯粹是对向 window. load 注册事件的替代方法。通过使用这个方法，可以在 DOM 载入就绪且能够读取并操纵时立即调用用户所绑定的函数，而大多数的 JavaScript 函数都需要在那一刻执行。

该函数有一个参数 fun，它会传递到这个 ready 事件处理函数中，可以给这个参数任意起一个名字，并因此可以不再担心命名冲突而放心地使用 $ 作为别名。

请确保在元素的 onload 事件中没有注册函数，否则不会触发 $(document). ready()事件。可以在同一个页面中无限次地使用 $(document). ready()事件。其中注册的函数会按照(代码中的)先后顺序依次执行，在 DOM 加载完成时运行的代码可以这样写：

```
$(document).ready(function(){
    //代码逻辑……
});
```

使用 $(document). ready()的简写，同时内部的 jQuery 代码依然使用 $ 作为别名，而不管全局的 $ 为何值。

```
jQuery(function($) {
    //这里继续使用$作为别名……
});
```

2. 事件处理

(1) bind 事件。bind(type,[data],fn)为每个匹配元素的特定事件绑定事件处理函数。bind()方法是用于往文档上附加行为的主要方式。所有 JavaScript 事件对象，比如 focus、mouseover 和 resize，都是可以作为 type 参数传递进来的。

jQuery 还提供了一些绑定这些标准事件类型的简单方式，比如 click()用于简化 bind('click')。这些事件还包括：blur、focus、focusin、focusout、load、resize、scroll、unload、click、dblclick、mousedown、mouseup、mousemove、mouseover、mouseout、mouseenter、mouseleave、change、select、submit、keydown、keypress、keyup、error。

任何作为 type 参数的字符串都是合法的，如果一个字符串不是原生的 JavaScript 事件名，那么这个事件处理函数会绑定到一个自定义事件上。这些自定义事件绝对不会由浏览器触发，但可以通过使用 trigger()或者 triggerHandler()在其他代码中手动触发。

当一个事件传到一个元素上，所有绑定在上面的针对哪个事件的处理函数都会触发。如果注册了多个事件处理函数，总是按照绑定的顺序依次触发。当所有绑定的事件处理函数执行完毕后，事件继续沿着普通的事件冒泡途径上浮。

事件处理函数中的 fn 参数接受一个回调函数，就像先前展示的那样。在这个事件处理函数内部，this 指向这个函数绑定的 DOM 元素。如果要让这个元素变成 jQuery 对象来使用 jQuery 的方法，可以把这个对象传入 $()并重新封装。

【例 6-8】 事件的使用。

```
<h2>This is a heading</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
<button type="button">Click me</button>
<script type="text/javascript">
$(document).ready(function(){          //载入函数
    $("button").click(function(){      //给按钮添加 bing 事件
        $("p").hide();                 //隐藏所有<p>元素
    });
});
```

以上代码在 ready()加载事件中给 button 对象绑定了 click 事件,事件执行后将隐藏所有的<p>元素,运行效果如图 6-6 和图 6-7 所示。

图 6-6　事件执行之前

图 6-7　事件执行之后

(2) unbind 事件。unbind([type],[fn])为 bind()函数的反向操作,从每一个匹配的元素中删除绑定的事件。如果没有参数,则删除所有绑定的事件。例如:

$("p").unbind();//把所有段落的所有事件取消绑定。

$("p").unbind("click"); //将段落的 click 事件取消绑定。

(3) one 事件。one(type,[data],fn)用于为每个匹配元素的特定事件绑定一个一次性事件处理函数。这个事件处理函数只会被执行一次。其他规则与 bind()函数相同,如下面的代码中在第一次单击的时候,会显示所有的文本,后面的执行将无效。

```
$("p").one("click", function(){
    alert($(this).text() );
});
```

(4) trigger 事件。trigger(type,[data])用于在每一个匹配的元素上触发某类事件。例如:

```
$("form:first").trigger("submit");     //提交第一个表单
$("p").click(function (event, a, b) {
    //出现一个普通的单击事件时,a 和 b 是 undefined 类型
    //如果用下面的语句触发,那么 a 指向 foo,而 b 指向 bar
} ).trigger("click", ["foo", "bar"]);
```

3. 事件切换

(1) hover 事件。hover(over,out)用于模仿悬停事件的方法,即鼠标光标移动到一个

对象上面后又移出这个对象。这是一个自定义的方法，它为频繁使用的任务提供了一种“保持在其中”的状态。

当鼠标光标移动到一个匹配的元素上面时，会触发指定的第一个函数。当鼠标光标移出这个元素时，会触发指定的第二个函数。而且会伴随着对鼠标是否仍然处在特定元素中的检测（例如，处在 div 中的图像），如果是，则会继续保持“悬停”状态，而不触发移出事件，示例代码如下：

```
$("tr").hover(
    function () {
        $(this).addClass("hover");
    } //表示鼠标移到元素上要触发的函数
    function () {
        $(this).removeClass("hover");
    }//表示鼠标移出元素要触发的函数
);
```

（2）toggle 事件。toggle(fn,fn2,[fn3,fn4,...])表示每次单击后依次调用函数。如果单击了一个匹配的元素，则触发指定的第一个函数；当再次单击同一元素时，则触发指定的第二个函数；如果有更多函数，则再次触发，直到最后一个。随后的每次单击都重复对这几个函数的轮番调用。可以使用 unbind("click")来解除该事件的调用，如例 6-9 中对表格的<td>执行样式进行切换。

【例 6-9】 toggle 事件示例。

```
<style type="text/css">
    .selected{ background-color:#eeeee;}
</style>
<table border="1">
    <tr>
        <td>Go to the store</td><td>Pick up dinner</td>
        <td>Debug crash</td><td>Take a jog</td>
    </tr>
</table>
<script type="text/javascript">
    $(document).ready(function(){                    //载入函数
        //对</td>执行样式切换
        $("td").toggle(
            function () {
                $(this).addClass("selected");        //添加样式类
            },
            function () {
                $(this).removeClass("selected");     //移除样式类
            }
        );
    });
</script>
```

4. 其他事件

jQuery 还提供丰富的其他事件，如焦点事件、键盘事件、鼠标事件等，表 6-11 列举了部分常用事件。

表 6-11　常用事件

事件名称	事件的原型结构	使用示例

聚焦	onfocus([fn])	$("document").ready(function(){
		　$("#name").focus();
		}); //当页面加载后将 id 为 name 的元素设置为焦点

失去焦点	onblur([fn])	$("input[type=text]").focus(function(){
		　this.blur();
		}); //无法使用文本框

按下键盘	keydown([fn])	$(window).keydown(function(event){
		　switch(event.keyCode) {
		　　//不同的按键可以做不同的事情
		　}
		}); //在页面内对键盘按键做出回应

按下并松开按键	keypress([fn])	
松开按键	keyup([fn])	
页面加载	load(fn)	$(window).load(　function() { alert("Hello!"); });
页面卸载	unload(fn)	$(window). unload (　function() { alert("Hello!"); });
鼠标单击	click([fn])	$("p").click(function() { $(this).hide(); });

鼠标上移	mouseover(fn)	$("p"). mouseover (
		　function() { alert($(this).html());
		}); //输出<p>的 HTML 内容

鼠标移开	mouseout(fn)	$("p").click(function() { $(this).hide(); }); //隐藏自己
鼠标双击	dbclick([fn])	$("p").dblclick(function(){alert("Hello World!"); });

文本/索引的改变	change([fn])	$("input[type='text']").change(function() {
		　//这里可以写些验证代码
		});

改变窗体大小	resize	$(window).resize(function(){
		　alert("Change Window!");
		});

页面滚动条发生变化	scroll	$(window).scroll(function() {
		/* ...do something... */});
		//当页面滚动条变化时执行的函数

文本被选中	select	$(":text").select(function() { /* ...do something... */}); //当文本框中文本被选中时执行的函数
提交表单	submit	$("form").submit(function() { return false;}); //阻止表单的提交

6.2.5 jQuery 动画

jQuery 提供了常用的动画效果，减少了编程过程中的动画编写难度，开发者不需要考虑不同浏览器的差异性，开发者也可以根据自己的要求自定义动画。

1. show()/hide()/toggle()

显示(show())和隐藏(hide())方法对于动画来说是最基本的效果，例 6-10 演示了对段落标记的隐藏和显示。

【例 6-10】 show()/hide()使用示例 1。

```
<script type="text/javascript">
    $(function() {
        //隐藏段落
        $("input:first").click(function() {
            $("p").hide();
        });
        //显示段落
        $("input:last").click(function() {
            $("p").show();
        });
    });
</script>
<input type="button" value="Hide">
<input type="button" value="Show">
<p>单击按钮,看看效果</p>
```

show()/hide()方法还可以接收参数。根据参数的不同，可以达到显示不同效果的目的，其语法如下：

```
show(duration,[callback]);
hide(duration,[callback]);
```

其中，duration 表示动画执行时间的长短，可以用与速度有关的字符串来表示，包括 slow、normal、fast，也可以是表示时间的整数(毫秒)。callback 是可选的回调函数，在动画完成之后执行。如例 6-11 中进行显示需要 500 毫秒，隐藏需要 300 毫秒，隐藏结束或调用回调函数，弹出消息“已隐藏”。

【例 6-11】 show()/hide()使用示例 2。

```
<script type="text/javascript">
    $(function() {
        //隐藏需要 300 毫秒
        $("input:first").click(function() {
            $("p").hide(300,function(){
                alert("已隐藏"); //执行回调函数
            });
        });

        //显示过程需要 500 毫秒
```

```
        $("input:last").click(function() {
            $("p").show(500);
        });
    });
</script>
<input type="button" value="Hide">
<input type="button" value="Show">
<p>单击按钮,看看效果</p>
```

toggle()方法用于切换元素的可见状态,如果被选元素可见,则隐藏这些元素;如果被选元素隐藏,则显示这些元素。语法如下:

```
$(selector).toggle(speed,callback,switch)
```

参数 speed、callback 同 show()/hide()方法中的对应参数。参数 switch 为布尔值,规定了 toggle 是否隐藏或显示所有被选元素。

【例 6-12】 toggle 使用示例。

```
<script type="text/javascript">
    $(document).ready(function(){
        $(".btn1").click(function(){
            $("p").toggle(true);
        });
    });
</script>
<body>
<p>This is a paragraph.</p>
<p style="display:none">This is another paragraph.</p>
<p>把 switch 参数设置为 false,可以隐藏所有段落。</p>
<button class="btn1">显示所有 p 元素</button>
</body>
```

2. fadeIn()和 fadeOut()

fadeIn()和 fadeOut()用于实现渐隐渐现的效果,其语法与 slow()和 hide()完全相同,语法如下:

```
fadeIn(duration, [callback]);
fadeOut(duration, [callback]);
```

【例 6-13】 fadeIn()和 fadeOut()使用示例。

```
<script type="text/javascript">
    $(function() {
        $("input:eq(0)").click(function() {
            $("img").fadeOut(3000);         //逐渐淡出
        });
        $("input:eq(1)").click(function() {
            $("img").fadeIn(1000);          //逐渐淡入
```

```
        });
    });
</script>
<img src="img.jpg">
<input type="button" value="Hide">
<input type="button" value="Show">
```

3. slideUp()和 slideDown()

slideUp()和 slideDown()可以实现拉幕效果，slideUp()实现向上拉幕，slideDown 实现向下降幕，如例 6-14 所示。

【例 6-14】 slideUp()和 slideDown()使用示例。

```
<script type="text/javascript">
    $(function () {
        //向上拉幕
        $("input:eq(0)").click(function () {
            $("div").add("img").slideUp(1000);
        });
        //向下降幕
        $("input:eq(1)").click(function () {
            $("div").add("img").slideDown(1000);
        });
    });
</script>
<input type="button" value="SlideUp">
<input type="button" value="SlideDown">
<br>
<div></div><img src="img.jpg">
```

4. 自定义动画

考虑到框架的通用性及代码文件的大小，jQuery 不能涵盖所有的动画效果，但它提供了 animate()方法，能够使开发者自定义动画，animate()方法在执行时遵从动画队列的两种规则。

(1) 一组元素上的动画效果。当在一个 animate()方法中应用多个属性时，动画是同时发生的；当以链式的写法应用动画方法时，动画是按顺序依次发生的。

(2) 多组元素上的动画效果。默认情况下，动画都是同时发生的。当以回调的形式应用动画方式时，动画是按照回调顺序发生的。

另外，在动画方法中，要注意其他非动画方法会插队，例如 css()方法要使非动画方法也按照顺序执行，这时需要把这些方法写在动画方法的回调函数中才可以解决问题。

animate()方法使用起来很简单，常用的形式为：

```
animate(params,[duration],[easing],[callback])
```

其中 params 为希望进行变幻的 CSS 属性列表，以及希望变化到的最终值。duration 为可选项，与 show()/hide()中的参数含义完全相同。easing 为可选参数，通常供动画插件

使用，用来控制动画节奏的变化。jQuery 中只提供了 linear 和 swing 两个值。callback 为可选的回调函数，在动画完成后触发，示例代码如例 6-15 所示。

【例 6-15】 animate 动画效果示例。

```
<style>
    div {background-color: #FFFF00;height: 40px;width: 80px;
    border: 1px solid #000000;margin-top: 5px;padding: 5px;
    text-align: center;}
</style>
<script type="text/javascript">
    $(function() {
        $("button").click(function() {
            $("#block").animate({opacity: "0.5",width: "80%",
                height: "100px",borderWidth: "5px",
                fontSize: "30px",marginTop: "40px",
                marginLeft: "20px"
            }, 1000);
        });
    });
</script>
<button id="go">执行动画</button>
<div id="block">动画显示区域</div>
```

animate()方法的功能强大，可以使用它来代替其他所有的动画方法。

(1) 用 animate()方法代替 show()方法

```
$("p").animate({height:"show",waidth:"show",opacity:"show"},200);
```

等同于

```
$("p").show(200);
```

(2) 用 animate()方法代替 fadeIn()方法

```
$("p").animate({opacity:"show"},200);
```

等同于

```
$("p").fadeIn(200);
```

(3) 用 animate()方法代替 slideDown()方法

```
$("p").animate({height:"show" },200);
```

等同于

```
$("p").slideDown(200);
```

（4）用 animate()方法代替 fadeTo()方法

```
$("p").animate({opacity:"0.6"},200);
```

等同于

```
$("p").fadeTo(200,0.6);
```

事实上，这些动画就是 animate()方法的一种内置了特定样式属性的简写形式。在 animate()方法中，这些特定样式的属性值可以为 show、hide 和 toggle，也可以是自定义数字。

6.2.6 DOM 操作

文档对象模型(document object model，DOM)是万维网联盟(the world wide web consortium，W3C)组织推荐的处理可扩展标志语言的标准编程接口，它是以面向对象方式描述的文档模型。使用 jQuery 获得的 MOM 对象包含了 DOM 对象的基本特性，同时又进行了扩展，所以要使用 jQuery 操作 DOM 对象前需要对其进行转换，在 jQuery 中可以使用关键字“$()”将普通 HTML DOM 对象转换为 jQuery DOM 对象，例如：

```
var jObj=$(document.getElementById("msg"));
```

以上代码中演示了如何获取 id="msg"的对象，然后将 DOM 对象转换为 jQuery 对象 jObj。

1. 查找节点

使用 jQuery 在文档树上查找节点非常容易。

（1）查找元素节点。获取元素节点的 jQuery 代码如下：

```
var $li=$("ul li:eq(1)");         //获取第二个<li>元素节点
var li_txt=$li.text();            //输出第二个<li>元素节点的 text
```

（2）查找属性节点。利用 jQuery 选择器查找到需要的元素后，就可以使用 attr()方法来获取它的各种属性的值。attr()方法的参数可以是一个，也可以是两个。当参数是一个时，则是要查询的属性的名字，例如：

```
var $para=$("p");                 //获取<p>节点
var p_txt=$para.attr("title");    //输出<p>元素节点属性 title
```

2. 创建节点

利用 jQuery 选择器能够快捷而轻松地查找到文档中的某个特定的元素节点，然后可以用 attr()方法来获取元素的各种属性的值。

使用 $(html)函数完成元素节点的创建，该函数会根据传入的 HTML 标记字符串创建一个 DOM 对象后返回，例如：

```
var $li_1=$("<li></li>");         //创建第一个<li>元素
var $li_2=$("<li></li>");         //创建第二个<li>元素
```

使用 append()或 appendTo()方法可将创建的元素插入文档中。

```
$("ul").append($li_1);
$("ul").append($li_2);          //添加到<ul>节点中
```

3. 插入节点

动态创建 HTML 元素通常需要插入文档中。让它成为这个文档的某个节点的子节点。前一节使用了一个插入节点的方法 append(),它会在元素内部追加新创建的内容。在 jQuery 中还提供了其他几种插入节点的方法,如表 6-12 所示。

表 6-12 插入节点的方法

方法	描述	示例
append()	向每个匹配的元素内部追加内容	HTML 代码: Hello jQuery 代码: $("p").append("jQuery") 结果: Hello:jQuery
appendTo()	将所有匹配的元素追加到指定的元素中	HTML 代码: Hello jQuery 代码: $("jQuery").appendTo("P") 结果: HellojQuery
prepend()	向每个匹配的元素内部前置内容	HTML 代码: Hello jQuery 代码: $("p").prepend("jQuery") 结果: jQueryHello
prependTo()	将所有匹配的元素前置到指定的元素中	HTML 代码: Hello jQuery 代码: $("jQuery").prependTo("p") 结果: jQueryHello
after()	在每个匹配的元素之后插入内容	HTML 代码: Hello jQuery 代码: $("p"). after ("jQuery") 结果: Hello jQuery

续表

方　法	描　　述	示　　例
insertAfter()	将所有匹配的元素插入指定的元素前	HTML 代码： Hello jQuery 代码： $("jQuery"). insertAfter ("P") 结果： 你好： Hello
before()	在每个匹配的元素之前插入内容	HTML 代码： Hello jQuery 代码： $("p"). before ("jQuery") 结果： jQuery Hello
insertBefore()	将所有匹配的元素插入指定的元素后	HTML 代码： Hello jQuery 代码： $("jQuery"). insertBefore ("p") 结果： Hello jQuery

4. 删除节点

jQuery 提供了两种删除节点的方法，即 remove()和 empty()。

(1) remove()。remove()方法用于从 DOM 中删除所有匹配的元素，例如：

```
$ ("ul li:eq(0)").remove();
```

该方法在获取第一个<li>元素节点后，将匹配的元素从网页中删除。

当某个节点用 remove()方法删除后，该节点所包含的所有后代节点将同时被删除。这个方法的返回值是一个指向已被删除的节点的应用，因此可以以后再使用这些元素。

(2) empty()。empty()方法不是删除节点，而是清空节点，它能清空元素中的所有后代节点。

```
$("ul li:eq(1)").empty();
```

该方法在获取第二个元素节点后再清空此元素里的内容。

5. 替换节点

jQuery 提供了 replaceWith()和 replaceAll()方法替换节点。

replaceWith()方法的作用是将所有匹配的元素都替换成指定的 HTML 或者 DOM 元素。

例如要将网页中的"<p title="Hellow jQuery!">Hellow jQuery? </p>"替换成"Hellow jQuery? "，可以使用如下的 jQuery 代码。

```
$("p").replaceWith("<strong>Hellow jQuery?</strong>");
```

也可以用 replaceAll()来实现，该方法与 replaceWith()方法的作用相同，只是颠倒了 replaceWith()操作，可以使用如下的 jQuery 代码实现同样的功能。

```
$("<strong>Hellow jQuery?</strong>").replaceAll("p");
```

6. 属性操作

在 jQuery 中，用 attr()方法可以获取和设置元素的属性，removeAttr()方法可以删除元素的属性。

(1) 获取元素值。

```
var d=$("div");                    //获取<div>节点
vartxt=d.attr("title");            //获取<div>元素节点的 title 属性
```

(2) 设置属性值。如果要设置<p>元素的 title 属性值，也可以使用同一个方法，不过需要传递两个参数，即名称和对应的值。

```
$("p").attr("title","you title");         //设置单个的属性值
```

jQuery 可以一次性为同一个元素设置多个属性，可以使用下面的代码来实现。

```
$("p").attr({"title": "标题内容","name":"pTest"});
```

以上代码将一个“名/值”形式的对象设置为匹配元素的属性。jQuery 中的很多方法都是用同一个函数实现获取(getter)和设置(setter)的功能，例如上面的 attr()方法既能设置元素属性的值，也能获取元素属性的值。类似的还有 html()、text()、height()、width()、val()和css()等方法。

(3) 删除属性。使用 removeAttr(attrName)方法删除属性，如以下代码删除<p>元素的 title 属性。

```
$("p").removeAttr("title");
```

7. 样式操作

(1) 追加样式。jQuery 中使用 addClass()方法来追加样式，如例 6-16 所示。

【例 6-16】 追加样式。

```
<!--样式-->
<style type="text/css">
    .oldStyle{ border:1px solid #000;width:200px;height:25px;}
    .newStyle {font-size:36px; font-style:italic; font-weight:bolder;}
</style>

<script type="text/javascript">
    $("#btnAdd").click(function () {
```

```
        $("p").addClass("newStyle"); //追加 newStyle 样式
    });
</script>
<div class="oldStyle">这里是原来的样式</div>
<p class="oldStyle">这里添加了新样式</p>
<input type="button" id="btnAdd" value="追加样式" />
```

以上代码中,当单击 id="btnAdd"的按钮时,则向所有的<p>标签追加 class 类,显示效果如图 6-8 所示。

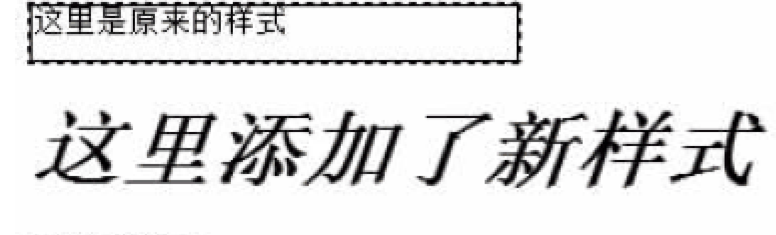


图 6-8 追加样式

(2) 移除样式。在例 6-16 中,为<p>元素追加了 newStyle 样式。此时<p>元素的 HTML 代码变为<p class="oldStyle newStyle">。这里添加了新样式</p>。如果要删除元素的样式,可使用 removeClass()方法,它的作用是从匹配的元素中删除全部或者指定的 class。

```
$("p").removeClass("high");          //删除<p>元素的 newStyle 样式
$("p").removeClass();                //删除<p>元素的所有样式
```

(3) 切换样式。jQuery 中使用 toggle()方法实现追加样式和移除样式的交换操作,语法如下:

```
$(选择器).toggle(function(){
    //显示元素
},function(){
    //隐藏元素
})
```

此处 toggle()方法的作用是交替执行两个函数,如果元素原来是显示的,则隐藏它;如果元素原来是隐藏的,则显示它。此时,toggle()方法主要是控制行为上的重复切换。

(4) 判断样式。hasClass()可以用来判断元素中是否含有某个 class,如果含有则返回 true,否则返回 false,如:

```
$("p").hasClass("newStyle");
```

以上代码判断<p>元素中是否含有 newStyle 的 class 样式。

8. 设置和获取 HTML 或文本

(1) html()。jQuery 中的 html()方法用于获取元素的 HTML 内容,该方法的功能类似于 JavaScript 中的 innerHTML 属性,可以用来读取或者设置某个元素中的 HTML 内容。该方法可以有一个参数,当给参数赋值时,其功能为设置元素的 HTML 内容。

```
var html=$("p").html();                   //获取<p>元素的 HTML 内容
$("p").html("<b>Hello world</b>");        //设置<p>元素的 HTML 内容为"<b>Hello
                                            world</b>"
```

(2) text()方法。text()方法类似于 JavaScript 中的 innerText 属性,可以用来读取或者设置某个元素中的文本内容。该方法可以有一个参数,当给参数赋值时,其功能为设置元素的文本内容。

```
var html=$("p").text ();          //获取<p>元素的文本内容
$("p").text("Hello world");     //设置<p>元素的 HTML 文本内容为"Hello world"
```

(3) val()方法。val()方法类似于 JavaScript 中的 value 属性,可以用来设置和获取元素的值。无论元素是文本框、下拉列表还是单选框,它都可以返回元素值。如果元素为多选,则返回一个包含所有选择值的数组。val()方法带一个参数时表示给对象设置值。

```
$("#name").val();                //获取 id="name"元素的文本值
$("#name ").val("jQuery");       //获取 id="name"元素的文本值为 jQuery
$(":checkbox").val();            //获取所有 type="checkbox"元素的选中值
$(":radio").val(["1"]);          //初始化 type=" radio"元素的选中值
```

6.2.7　解决冲突

在 jQuery 中,$ 是 jQuery 的别名,所有使用 $ 的地方也都可以使用 jQuery 来替换,如 $('#id')等同于 jQuery('#id') 的写法。但很多时候自定义 $(id)方法来获取一个元素,或者其他的一些 JavaScript 类库。如:Prototype 也定义了 $ 方法,如果同时把这些内容放在一起就会引起变量方法定义的冲突,jQuery 专门提供了方法用于解决此问题,即通过调用 $.noConflict()向该库返回控制权。下面以引入两个库文件 jquery.js 和 prototype.js 为例来进行说明。

1. jquery.js 在 prototype.js 之后进行引入

```
<!--引入 prototype  -->
<script src="prototype.js" type="text/javascript"></script>
<!--引入 jQuery  -->
<script src="jquery.min.js" type="text/javascript"></script>
```

在这种情况下,选择器代码如果为 $('#id').hide(),则 $ 代表的永远是 jQuery 中定义的 $ 符号,但可以写成 jQuery('#id').hide()来解决此问题。如果要使用 Prototype 中的 $,则应使用 jQuery.noConflict()方法。解决方法有四种。

(1) 将 $ 的控制权让给 Prototype,示例代码如例 6-17 所示。

【例 6-17】 jquery.js 在 prototype.js 之前进行引入的冲突解决。

```
<div id="divJquery">
    jQuery 操作:<br/>
    jquery.js 在 prototype.js 之前进行引入的冲突解决
</div>
<div id="divPrototype">Prototype 操作区域</div>
<script type="text/javascript">
    //将变量$的控制权让给 prototype.js
```

```
    jQuery.noConflict();

    //使用 jQuery
    jQuery(function(){
        jQuery("#divJquery").click(function(){
            //弹出 jQuery 操作对象的 text 内容
            alert(jQuery(this).text());
        });
    });

    //使用 prototype
    $("divPrototype").style.display='none';
</script>
```

例 6-17 中，程序运行时 id＝"divPrototype"的标签会隐藏，单击标签 id＝"divJquery"时，会用 alert()弹出执行后的结果信息，程序的运行结果如图 6-9 所示。

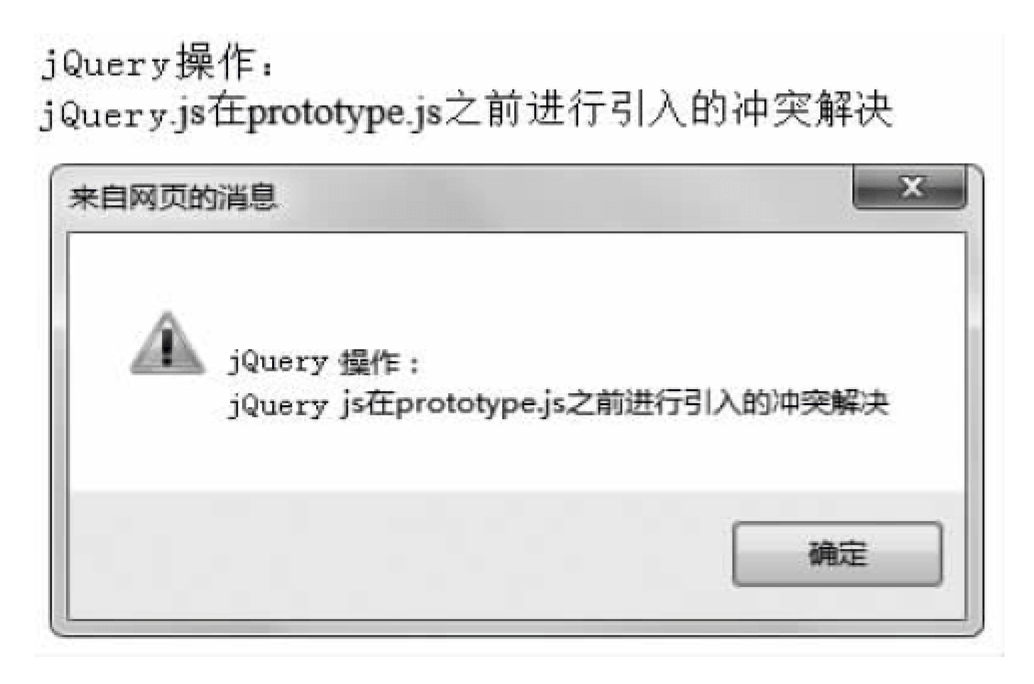


图 6-9　jQuery 冲突解决运行结果 1

（2）自定义一个 jQuery. noConflict()比较短的对象，使用的时候直接调用较短的对象来代替 jQuery 库，示例代码如下：

```
<div id="divJquery">jQuery 操作区域</div>
<div id="divPrototype">prototype 操作区域</div>
<script type="text/javascript">
    //自定义一个比较的 jQuery 库对象
    var $jq=jQuery.noConflict();

    //使用 jQuery
    $j(function(){
        $jq("#divJquery").click(function(){
            alert($jq(this).text());
        });
    });

    //使用 prototype
    $("divPrototype").style.display='none';
</script>
```

(3) 给 jQuery(function()) 函数指定参数为 $，这样就可以继续在该函数内使用 $ 作为 jQuery 对象，示例代码如下：

```
<div id="divjQuery">jQuery 操作区域</div>
<div id="divPrototype">prototype 操作区域</div>
<script type="text/javascript">
    //将变量$的控制权让给 prototype.js
    jQuery.noConflict();

    //使用 jQuery,在方法体内保留$的使用控制
    jQuery(function($){
        $("#divJquery").click(function(){
            alert($(this).text() );
        });
    });

    //使用 prototype
    $("divPrototype ").style.display='none';
</script>
```

(4) 使用匿名函数方式解决，在匿名函数传递参数时，实参使用 jQuery，示例代码如下：

```
<div id="divjQuery">jQuery 操作区域</div>
<div id="divPrototype">prototype 操作区域</div>
<script type="text/javascript">
    //将变量$的控制权让给 prototype.js
    jQuery.noConflict();
    //定义匿名函数并设置形参为$
    (function($){                //匿名函数内部的$均为 jQuery
        $(function(){
            //继续使用 $方法
            $("#divJquery").click(function(){
                alert($(this).text());
            });
        });
    })(jQuery);                  //执行匿名函数且传递实参 jQuery

    //使用 prototype
    $("divPrototype").style.display='none';
</script>
```

2. jquery.js 在 prototype.js 之前进行引入

在这种情况下，如果我们直接写 $('#id')，则 $ 此时代表的 prototype.js 中定义的 $ 符号。如果想要调用 jquery.js 中提供的各种功能，只能用全称写法 jQuery('#id') 才能实现，代码如例 6-18 所示。

【例 6-18】 jquery.js 在 prototype.js 之前进行引入的冲突解决。

```
<!--引入 jQuery  -->
<script src="jquery.min.js" type="text/javascript"></script>
<!--引入 prototype  -->
<script src="prototype.js" type="text/javascript"></script>
<body>
<div id="divJquery">
    jquery.js 在 prototype.js 之前进行引入的冲突解决<br/>
    执行结果：jQuery 操作的对象弹出对话框<br/>
    Prototype 执行后，字体变大
</div>
<div id="divPrototype">Prototype 操作区域</div>
<script type="text/javascript">
    jQuery.noConflict();                    //将变量$的控制权让给 prototype.js
    (function($){                           //定义匿名函数并设置形参为$
        $(function(){                       //匿名函数内部的$均为 jQuery
            $("#divJquery").click(function(){            //继续使用 $方法
                alert($(this).text());
            });
        });
    })(jQuery);                             //执行匿名函数且传递实参 jQuery

    //使用 prototype
    $("divPrototype").style.border="5px solid #000";    //设置边框
</script>
</body>
```

例 6-18 在运行后，id="divPrototype"的标签会生成 2 个像素的黑色边框，单击标签 id="divPrototype"，会弹出提示信息，程序的运行结果如图 6-10 所示。

jquery.js在prototype.js之前进行引入的冲突解决
执行结果：jQuery操作的对象弹出对话框
Prototype执行后，字体变大。
Prototype操作区域

图 6-10　jQuery 冲突解决运行结果 2

6.2.8　编写插件

插件编写的目的是给已经有的一系列方法或函数做一个封装，以便在其他地方重复使用，方便后期的维护。jQuery 除了提供一个简单、有效的方式管理元素以及脚本外，还提供了一种机制：即给核心模块增加自己的方法和额外的功能。通过这种机制，jQuery 允许用户创建属于自己的插件，从而提高开发效率。

1. 类级别的插件开发

对类级别的插件开发最直接的理解就是给 jQuery 类添加类方法，可以理解为添加静态方法。典型的例子就是 $.ajax() 函数，将函数定义于 jQuery 的命名空间中。关于类级别的插件开发可以采用如下几种形式进行扩展。

(1) 添加一个新的全局函数。添加一个全局函数，定义如下：

```
jQuery.foo=function() {
    alert(这是一个自定义全局函数的例子);
};
```

调用方法可以这样写：

```
jQuery.foo();
```

或者

```
$.foo();
```

(2) 增加多个全局函数。如果要同时添加多个全局函数，可采用如下定义：

```
jQuery.foo=function() {
    alert(这是一个自定义全局函数的例子); };
};
jQuery.bar=function(param) {
    alert("这是一个自定义全局函数的例子,参数为"+param);
};
```

调用时与一个函数的调用方法相同。

```
jQuery.foo();
jQuery.bar();
```

或者

```
$.foo();
$.bar('bar');
```

(3) 使用 jQuery.extend(object)方法。我们还可以使用 jQuery.extend(object)方法来扩展并定义新的全局函数，表达形式如下：

```
jQuery.extend({
    foo: function () {
        alert(这是一个自定义全局函数的例子);
    },
    bar: function (param) {
        alert("这是一个自定义全局函数的例子,参数为"+param);
    }
});
```

调用的时候可以这样写：

```
jQuery.foo();
```

或者

```
$.foo();
```

(4) 使用命名空间。虽然在 jQuery 命名空间中,禁止使用大量的 JavaScript 函数名和变量名。但是仍然难以避免某些函数或变量名将与其他 jQuery 插件冲突,因此用户最好将方法封装到另一个自定义的命名空间中,可以避免命名空间内函数的冲突。

```
jQuery.myPlugin={
    foo: function () {
        alert('这是一个自定义全局函数的例子');
    }, bar: function (param) {
        alert('这是一个自定义全局函数的例子,参数为'+param);
    }
};
```

采用新命名空间的函数仍然是全局函数,调用时需要指定命名空间,调用的方法为:

```
$.myPlugin.foo();
$.myPlugin.bar('bar');
```

2. 对象级别的插件开发

对象级别的插件开发有两种形式。

形式 1:此种形式通过 fn. extend()函数的方法来定义插件的名称,结构定义如下:

```
(function ($) {
    $.fn.extend({
        pluginName: function (opt, callback) {
            //插件的逻辑代码在这里实现
        }
    })
})(jQuery);
```

形式 2:此种形式通过直接指定插件名称来实现插件,结构定义如下:

```
(function ($) {
    $.fn.pluginName=function () {
        //插件的逻辑代码在这里实现
    };
})(jQuery);
```

上面两种实现方式均定义了一个 jQuery 函数,形参是 $。函数定义完成之后,把 jQuery 这个实参传递进去,这样我们在写 jQuery 插件时,就可以在插件内部使用 $ 这个别名,而不会与 prototype 等其他 JavaScript 库起冲突。下面将详细分析各种情况的使用示例。

(1) 在 jQuery 名称空间下声明一个名字。这里举例说明一个单一插件的实现表达。如果你准备同时开发多个插件,那么需要声明多个函数名字。通常当在编写一个插件时,力求仅使用一个名字来包含它的所有内容。比如要创建一个分页的插件,将其命名为

Paging，则该插件的示例代码为：

```
$.fn.Paging=function() {
    //分页代码的逻辑写在这里
};
```

Paging 插件的调用方式为：

```
$('#pageDiv').Paging();
```

(2) 设置 options 参数控制插件的行为。在开发插件时，往往会向插件内部提供一定的参数，通过这个参数可以控制插件的行为，比如为 Paging 插件指定当前分页的索引页（参数 pageIndex）和每页显示的数据条数（参数 pageSize）。通常的做法是，将所有的参数封装在一个 JSON 对象中，这个对象的名称为 options，最后将对象传递给插件函数。例如：

```
//定义名称为 Paging 的插件
$.fn.Paging=function(options)
{
    //设置参数的默认值
    var defaults={
        pageIndex:1,              //当前索引页为 1
        pageSize:25               //每页显示的记录条数为 25
    };

    //实现插件的参数扩展
    var opts=$.extend(defaults, options);

    //插件的逻辑代码在这里实现
};
```

Paging 插件的调用方式为：

```
$('#pageDiv').Paging({
    pageIndex: 2,            //当前索引页为 1
    pageSize: 30             //每页显示的记录条数为 30
});
```

(3) 暴露插件的默认设置。为减少插件的使用者用较少的代码覆盖默认参数，可以通过暴露插件的默认的方式设置 options 参数，以 Paging 插件为例，示例代码如下：

```
//定义 Paging 插件
$.fn.Paging=function (options) {
    //扩展参数 options
    //如果有了参数默认值,则 options 中的参数则不会覆盖默认的参数,减少 options 参数的
      设置
```

```
    var opts=$.extend(
        {},
        $.fn.Paging.defaults,       //默认参数
        Options                     //形参,需要传入
    );

    //插件的逻辑代码在这里实现
};

//设置插件参数的默认选项值
$.fn.Paging.defaults={
    //这里实现参数的默认值设置
    pageSize : 25                   //指定默认参数,每页显示记录条数为 25
};
```

插件的调用。由于使用了默认值覆盖参数的方法,在程序调用时,只需要在 $(document).ready()函数中调用一次插件的默认方法,以后插件的使用均不需要再重复指定默认值中的参数。

如果这样调用插件:

```
$('#pageDiv').Paging();
```

调用后,分页插件会每页显示 25 条记录。

如果在 $(document).ready()函数中这样调用了插件:

```
$(document).ready(function(){
    //设置分页插件的默认值
    $.fn.Paging.defaults.pageSize=30;

    //调用分页插件
    $('#pageDiv').Paging();
});
```

调用后,分页插件会每页显示 30 条记录。

还可以通过传递配置参数给插件方法来覆盖默认设置,示例如下:

```
$('#pageDiv').Paging({
    pageIndex: 2,           //当前索引页为 1
    pageSize: 30            //每页显示记录条数为 30
});
```

(4) 公开插件函数来提高插件的扩展性。如果想让开发的插件具有更好的灵活性,允许其他使用者进行功能扩展,除了可向插件传递参数外,还可以使用公开部分函数的方式来提升功能性。比如给 Paging 插件提供一个用于呈现分页控件 HTML 内容的函数 OutputHTML,示例代码如下:

```
//定义插件 Paging
$.fn.Paging=function (options) {
    //扩展选项
    var opts=$.extend(
        {},
        $.fn.Paging.defaults,       //默认参数
        Options                     //形参,需要传入
    );

    //调用公开函数 OutputHTML,输出具体的呈现内容
    return $this.html($.fn.Paging.OutputHTML(options.Data));
};

//定义公开函数
$.fn.Paging.OutputHTML=function(data) {
    //这里不用实现真正的代码逻辑,具体的实现逻辑由调用者来实现
    return "";
};

//设置插件参数的默认选项
$.fn.Paging.defaults={
    //这里实现参数的默认值设置
    pageSize: 25  //指定默认参数,每页显示记录条数为 25
};
```

调用以上代码只需实现公开的函数部分即可,比如如下的调用示例。

```
//实现插件的公开函数
$('#pageDiv'). OutputHTML=function (data) {
    var html="";
    //具体是分页呈现逻辑在这里实现
    return htm;
};

//调用分页插件
$('#pageDiv').Paging();
```

(5) 保持私有函数的私有性。有时用户不希望插件中的所有函数都能被公开访问,这就需要对函数的可访问性做一定的设置,通过这种方式的处理,可以根据插件的序号少暴露插件的函数信息。

比如用户希望在插件中能实现输出调试结果,可以在一个闭包的方法中定义此私有方法,示例代码如下:

```
(function($) {
    //定义插件 Paging
    $.fn.Paging=function(options) {
        OutputLogs(this);          //输出调试结果
```

```
        //分页插件的逻辑实现
    };

    function OutputLogs ($obj) {
        if(window.console && window.console.log)
        window.console.log('Paging 插件输出内容：'+$obj.html());
    };

    //其他逻辑
})(jQuery);
```

由于 OutputLogs 方法定义在特定环境中，闭包外部无法访问进来，因此可以实现函数的私有访问性。

3. 编写插件的注意事项

(1) 插件的推荐命名方法为：

```
jQuery.[插件名].js
```

(2) 所有的对象方法都应当附加到 jQuery. fn 对象上面，而所有的全局函数都应当附加到 jQuery 对象本身。

(3) 在插件内部，this 指向的是当前通过选择器获取的 jQuery 对象，而不像一般方法那样，内部的 this 指向的是 DOM 元素。

(4) 可以通过 this. each 来遍历所有的元素。

(5) 所有方法或函数插件，都应当以分号结尾，否则压缩的时候可能会出现问题。为了更加保险，可以在插件头部添加一个分号(;)，以免它们的不规范代码给插件带来影响。

(6) 插件应该返回一个 jQuery 对象，以保证插件的可链式操作。

(7) 避免在插件内部使用 $ 作为 jQuery 对象的别名，而应当使用完整的 jQuery 来表示，这样可以避免冲突。

jQuery 的插件分为封装对象方法、封装全局函数两种类型。

(1) 封装对象方法的插件，即给 jQuery 对象添加方法。

(2) 封装全局函数的插件(类级别开发)，即给 jQuery 添加新的全局函数，相当于给 jQuery 类本身添加方法，与调用 jQuery 其他函数的使用方法相同。

【例 6-19】 添加一个 jQuery 插件，使其显示提示框。

(1) 新建一个名为 jQuery. JQueryPlugInTest. js 的 JavaScript 文件，并添加下面的代码：

```
//定义插件,实现两个数的相加
jQuery.add=function (a,b) {
    return a+b;
};
```

(2) 新建一个名为 testPlugIn. html 的 HTML 页面。

(3) 打开 testPlugIn. html 页面，添加对 jQuery. JQueryPlugIntest. js 文件的引用，并添加调用新插件的函数，代码如下：

```
<script type="text/javascript" src="Scripts/jQuery.min.js"></script>
<script type="text/javascript" src="jQuery.JQueryPlugInTest.js"></script>
<script type="text/javascript">
    function test() {
        //调用插件 jQuery.JQueryPlugInTest.js 中的函数
        var a=1,b=2;
        var rv=($.addPlugin(1,2));
        alert(a.toString()+"+"+b.toString()+"="+rv);     //输出结果
    }
</script>
```

(4) 在<body></body>标签中添加"测试"按钮，代码如下：

```
<input type="button" onclick="test()" value="测试 jQuery 插件" />
```

(5) 单击"测试 jQuery 插件"按钮，界面如图 6-11 所示。



图 6-11　jQuery 插件运行示例

6.3　第三方插件及使用方法

由于 jQuery 支持灵活的插件开发，近几年来，jQuery 开发爱好者不断开发出各种优秀的第三方 jQuery 插件，通常按照如下五种不同的类型进行归类。

1. UI 类

UI 类的插件用于实现 UI 界面的美化，内容丰富，包括背景插件、对话框和灯箱插件、筛选及排序插件、反馈插件、弹出层插件、悬停插件、布局插件、图表插件、加载插件、圆边插件、滚动插件、标签插件、文本链接插件、工具提示插件、网络类型插件等，图 6-12 所示展示了酷炫的弹出层插件运行效果。

2. 输入类

输入插件用于实现各种数据的快速输入，常见输入类插件有：颜色拾取器插件、定制和风格插件、日期和时间插件、拖放插件、通用输入插件、自动完成插件、密码插件、投票率插件、搜索插件、选择框插件、快捷键插件、触摸插件、上传插件、验证插件等，图 6-13 所示为输入插件的运行效果。

3. 媒体类

媒体类控件用于实现个性化的媒体展示，包括音频和视频插件、幻灯片和轮播图插件、



图 6-12　弹出层插件运行效果

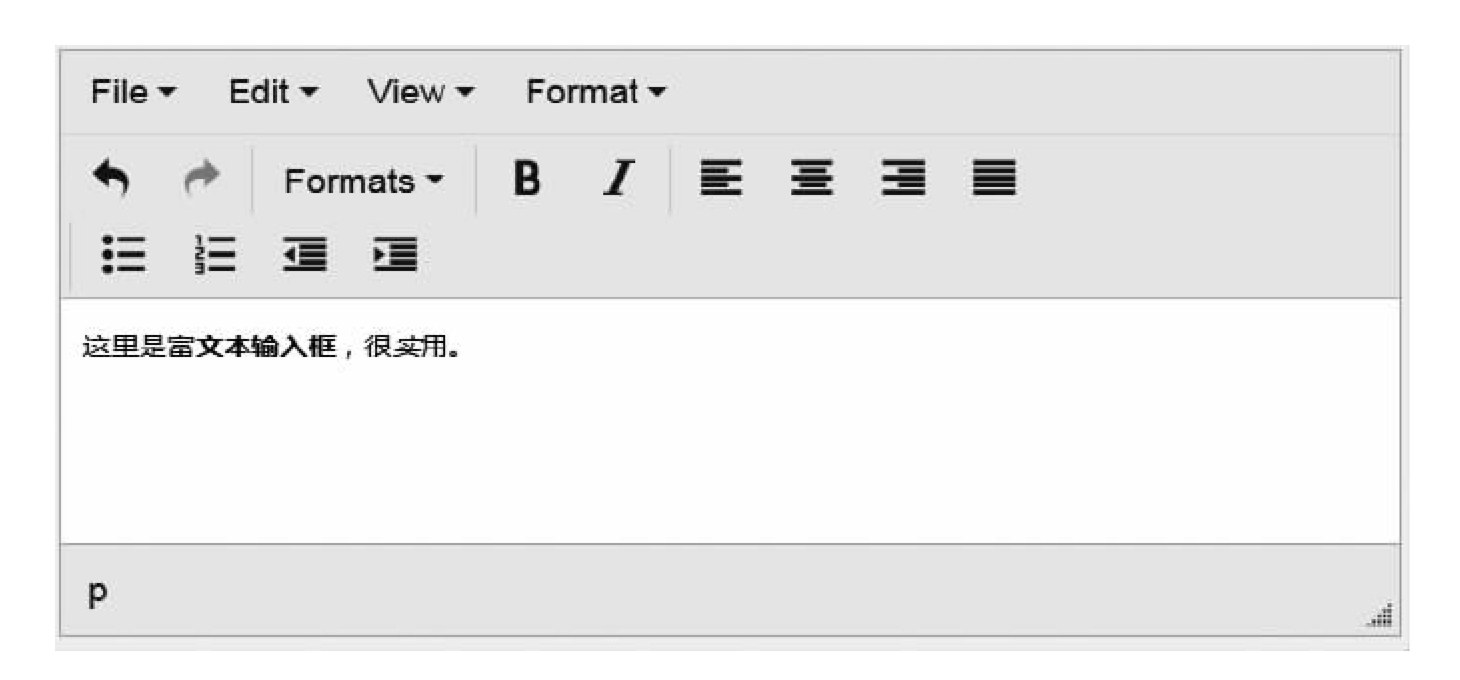


图 6-13　输入插件的运行效果

图片展示插件、图像插件、地图插件、滑块和旋转插件、Tabs 插件等，图 6-14 所示为幻灯片和轮播图插件的运行效果。

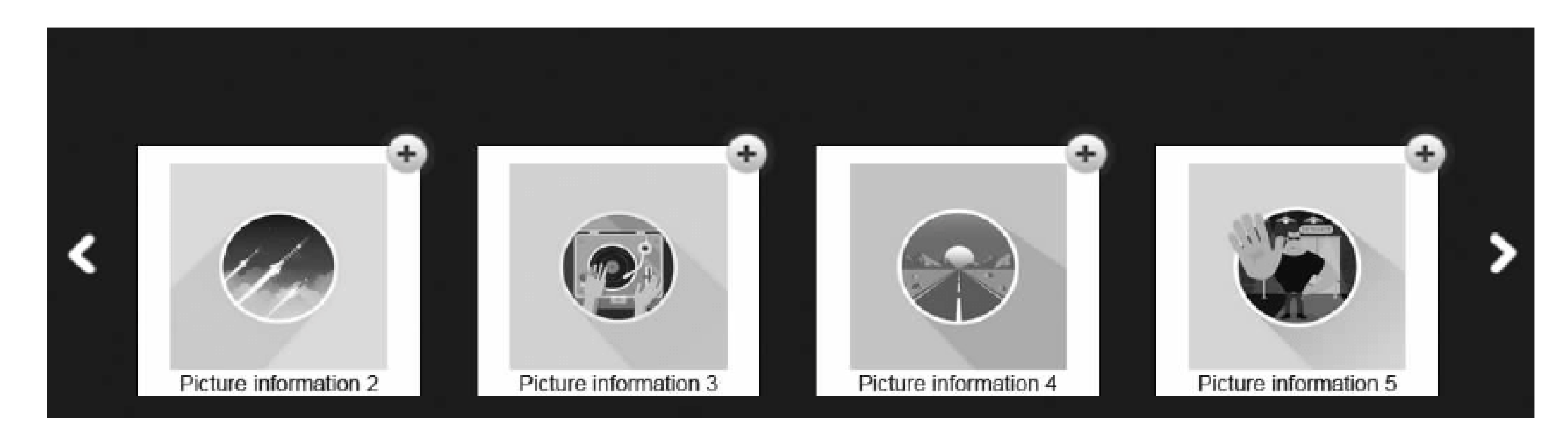


图 6-14　幻灯片和轮播图插件的运行效果

4. 导航类

UI 中导航的展示方式有很多种，常见的导航插件包括水平导航插件、垂直导航插件、文件树插件、分页插件、手风琴菜单插件等，图 6-15 所示为文件树插件的运行效果。

5. 其他类

除了以上四种大类插件外，还有一些适应性很强的插件，如动画效果、浏览器调整插件、移动插件、独立的部件插件、杂项插件、游戏插件等，图 6-16 所示为游戏插件的运行效果。



图 6-15　文件树插件的运行效果

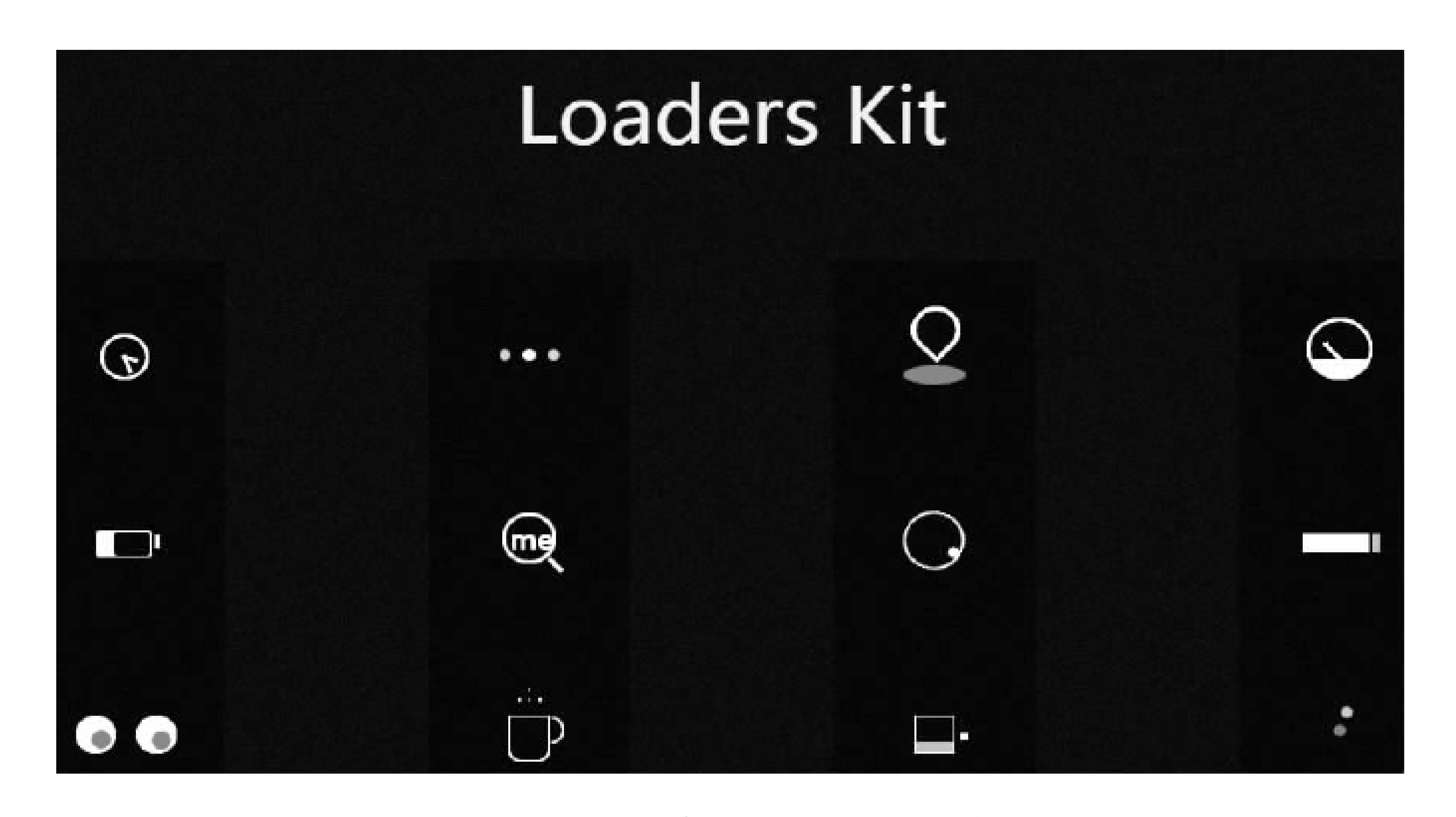


图 6-16　游戏插件的运行效果

尽管插件种类繁多，但都属于功能强大、使用简单的类型。本节将以输入类插件——校验控件（formValidator）和日期插件（My97DatePicker）为例来说明其使用方法。

6.3.1　校验控件 formValidator

在开发基于 B/S 架构体系风格的应用系统时，必须对用户输入的内容进行合法性验证，常见的验证有非空验证、长度验证、内容合法性验证（如数字、E-mail 格式等），以及业务逻辑验证（如系统中注册用户不能相同）。

Web 前端开发者（JavaScript 开发者）往往需要编写大量的 JavaScript 代码来进行元素校验，而这些验证在平时开发中不停地重复书写。一般要判断的表单元素比较多，开发过程就显得枯燥无味且浏览器之间兼容性差。

jQuery formValidator 属于输入类的插件，它的使用方法较简单，只需要进行简单的配置，无须编写代码就能实现表单的检验，使开发者重点关注业务逻辑的实现，无须关心验证的实现过程。它包括常规检验功能和可扩展校验功能。针对每个表单元素，只需要写一行配置信息就能完成校验。而这些配置信息无须写入表单元素，实现了 JavaScript 代码和 HTML 代码的分离且具备跨浏览器的能力。

1. 获取 formValidator

(1) 官方下载和在线演示地址：http://www.formvalidator.net/。

(2) 使用百度搜索 formValidator 获取 JavaScript 文件。

2. jQuery formValidator 插件的 API

目前 formValidator 支持五种大的校验方式：inputValidator（针对 input、textarea、select 控件的字符长度、值范围、选择个数的控制）、compareValidator（提供 2 个对象的比较，目前可以比较字符串和数值型）、ajaxValidator（通过 ajax 到服务器上做数据校验）、regexValidator（提供可扩展的正则表达式库）、functionValidator（提供可扩展函数库来做校验），每种校验方式支持的控件类型如表 6-13 所示。

表 6-13 formValidator 校验控件的五种校验方式与 HTML 控件之间的关系对照表

校验方式	text	radio	checkbox	file	password	textarea	select-one
inputValidator	√	√	√	√	√	√	√
compareValidator	√			√	√	√	
ajaxValidator	√			√	√	√	√
regexValidator	√			√	√	√	
functionValidator	√	√	√	√	√	√	√

目前插件提示错误有两种模式为 showword 和 showalert，即文字提示和窗口提示，下面的五种验证方式，针对 showalert 这种模式并不都是必需的，表 6-14 分别罗列出了全局初始化和五种校验方式公开的属性。

表 6-14 全局初始化及检验方式公开的属性

属性	属性名称	默认值	提示框	详细解释
validatorGroup	校验组	"1"	√	一个页面的控件可以分成多个组，分开校验
empty	确定是否可以为空	false	√	
autoModify	输入错误并离开焦点时，自动修复错误	true	√	先给出提示然后自动修复，目前只支持 text、file、textarea 三种类型
onEmpty	空值时的提示	"输入内容为空"		可以为空，为空时有提示。为空时即不显示内容
onShow	显示时的提示	"请输入内容"		为空时不显示内容
onFocus	获得焦点的提示	"请输入内容"		为空时不显示内容
onCorrect	输入正确后的提示	"输入正确"		当焦点离开控件的时候，如果输入正确将出现该提示。为空时不显示内容
tipID	显示错误的容器 ID	表单 ID+"Tip"		如果不自动构建提示层，表示提示层的 ID 号；如果自动构建提示层，表示提示层相对的目标控件

续表

属　性	属性名称	默　认　值	提示框	详细解释
tipCss	自动构建的提示层样式	"left":"10px";"top":"1px";"height":"20px";"width":"250px"		主要用于定位自动构建的提示层
forceValid	强制输入的值必须有效	true	√	确定是否把一个全角字符当作2个长度的参数
ajax	提交服务器	true	√	确定 ajaxValidator 是否把该表单提交给服务器
defaultValue	默认值	null	√	针对所有 input 和 select 表单。如果不设置就保持原值，一旦设置就设为默认值
triggerEvent	默认值	blur	√	当前支持 2 种属性值。 blur：失去焦点的时候触发。 change：当输入框里的值发生改变时触发
trimValue	默认值	false	√	当输入前后带空格的字符时，确定是否自动把空字符去掉

（1）inputValidator 校验控件的属性如表 6-15 所示。

表 6-15　inputValidator 校验控件的属性

属　性	属性名称	默认值	详细解释
type	比较类型	"size"	（对 select 无效） "size"：表示比较长度，为默认值。 "number"：数值型比较。 "string"：字符型比较。 "date"：短日期类型。 "datetime"：长日期类型
min	最小长度/值	0	默认数值型。如果进行字符比较，请输入字符型。 对于 select-one 而言，inputValidator 里的参数 min 和 max 表示选择的索引号范围；对于 select-multiple 而言，inputValidator 里的参数 min 和 max 表示选择的个数
max	最大长度/值	99999999999	同上
onError	发生错误时的提示	"输入错误"	为空时不显示内容
onErrorMin	比 min 属性小时的提示	null	当用户输入的值比 min 属性小时进行错误提示
onErrorMax	比 max 属性大时的提示	null	当用户输入的值比 max 属性大时进行错误提示
empty	确定控件文本值是否允许两边为空	两边都允许出现空	leftEmpty：表示左边是否允许为空，默认值为 true。 rightEmpty：表示右边是否允许为空，默认值为 true。 emptyError：出现该错误时提示。默认值为 null，表示利用 onError 属性来提示错误

（2）regexValidator 校验控件的属性如表 6-16 所示。

表 6-16　regexValidator 校验控件的属性

属　性	属 性 名 称	默认值	详 细 解 释
regExp	正则表达式或表达式数组	""	采用的是显式构造函数： new RegExp("pattern"[,"flags"])；由于 JavaScript 中'\'被用作转义字符，所以在使用显式构造函数构造实例对象的时候，需要使用'\\'代替'\'
param	附加参数	"i"	g：代表可以进行全局匹配。 i：代表不区分大小写进行匹配。 m：代表可以进行多行匹配。 以上参数可以任意组合，当然也可以不加参数
compareType	比较类型	"||"	"||"表示或的关系，"&&"表示并列
dataType	数据类型	"string"	"string"表示自己写的表达式；"enum"表示枚举名，字符串数组类型。具体请见 demo3. htm。 可以自己进行修改、添加 formValidatorRegex. js 里的枚举项目名和表达式
onError：	发生错误的提示	"输入错误"	为空时不显示内容

（3）ajaxValidator 校验控件的属性如表 6-17 所示。

表 6-17　ajaxValidator 校验控件的属性

属　性	属 性 名 称	默认值	详 细 解 释
type	请求的类型	"GET"	"POST" 或 "GET"
url	发送到的 URL 地址	""	在服务器端，可以通过 name 为 clientid 获取触发验证的控件 ID 名称
dataType	返回的数据类型	"html"	XML、HTML、Script、JSon、Text
timeout	超时设置	999	
data	数据	""	
async	是否以异步的方式发送	true	
success	当请求成功时调用的函数	null	
processData	自动处理返回的数据为字符串	true	在默认的情况下，如果 data 选项传入的数据是一个对象而不是字符串，将会自动地被处理和转换成一个查询字符串
complete	当请求完成时调用的函数	null	
beforeSend	当请求前时调用的函数	null	有一个参数，跟 $. ajax 里的 beforeSend 参数一样
buttons	单击“提交”按钮对应的 jQuery 对象	null	当你触发了 ajax 校验，buttons 里对应的按钮(组)就会变为灰色，一直等待服务器返回数据为止
error	当请求失败时调用的提示	"请求失败"	你可以自己定义这个错误，在 error 里自动显示出来。为空时不显示
onWait	正在校验的提示	"正在等待服务器返回数据"	

续表

属　性	属 性 名 称	默认值	详 细 解 释
onError	校验没有通过的提示	"服务器校验没有通过"	

注意：

① 在提交的时候自动追加 clientID 参数，表示触发校验的控件 ID。

② 在提交的时候自动追加 rand 参数，这是个随机数。

③ 几乎所有的属性跟 $.ajax() 的属性一样，请参考 $.ajax() 函数的帮助。

(4) functionValidator 校验控件的属性如表 6-18 所示。

表 6-18　functionValidator 校验控件的属性

属　性	属 性 名 称	默认值	返回值的解释
fun	外部函数名()。 参数 1：元素的值。 参数 2：元素对象	默认当作处理过程	true/false：校验成功/失败。 字符串：校验失败，返回值当作自定义错误。 无：处理过程
onError	发生错误的提示	"输入错误"	函数返回 false 的时候，显示该错误信息

(5) 在使用 formValidator 前需对其进行初始化，初始化 formValidator 的函数如表 6-19 所示。

表 6-19　初始化 formValidator 的函数

<table>
<tr><th>函　数　名</th><th>函 数 说 明</th></tr>
<tr><td>$.formValidator.initConfig</td><td>参数为配置类型。
<table>
<tr><th>属　性</th><th>默认值</th><th>说　明</th></tr>
<tr><td>validatorGroup</td><td>"1"</td><td>确定要针对哪个组进行配置</td></tr>
<tr><td>formID</td><td>""</td><td>要自动注册 pageIsValid 函数的表单 ID 号</td></tr>
<tr><td>alertMessage</td><td>false</td><td>确定是否弹出窗口</td></tr>
<tr><td>autoTip</td><td>false</td><td>确定是否自动构建提示层</td></tr>
<tr><td>errorFocus</td><td>true</td><td>确定发生错误的时候，第一个出错控件是否获得焦点</td></tr>
<tr><td>forceValid</td><td>true</td><td>确定是否一直到输入正确才允许离开焦点</td></tr>
<tr><td>wideWord</td><td>true</td><td>确定是否把一个全角字符当作 2 个长度</td></tr>
<tr><td>onSuccess</td><td>null</td><td>该组校验通过后的回调函数，返回 false 则阻止表单的提交</td></tr>
<tr><td>submitOnce</td><td>false</td><td>确定校验通过后，是否使所有的“提交”按钮变灰色</td></tr>
<tr><td>submitAfterAjaxPrompt</td><td>当前有数据正在进行服务器端校验，请稍候</td><td>参数 1：一个校验没有通过的错误信息。
参数 2：一个校验没有通过的元素对象。
参数 3：所有的错误信息数组，可以通过 $.map 来遍历</td></tr>
<tr><td>onError</td><td>null</td><td>该组校验失败后的回调函数，有两个参数</td></tr>
<tr><td>debug</td><td>false</td><td>是否处于调试模式。true：不提交表单</td></tr>
</table>
</td></tr>
</table>

续表

函　数　名	函 数 说 明
$.formValidator.pageIsValid	有一个参数，不是配置类型。validatorGroup 表示要针对哪个组进行验证
$.formValidator.isOneValid	有一个参数，为当时设置验证的表单元素 ID。返回是否校验成功的信息
$.formValidator.setFailState	用法为 function("tipid","显示的信息")。 在 showword 模式下，如果额外校验没有通过，可以通过它来设置成失败信息和状态
$.formValidator.getLength	用法为 function("表单元素 id")。 checkbox 或 radiobutton 表示(同组)选择选项的个数。 对于单选项，表示选择索引的值。 对于多选项，inputValidator 控件中的参数 min 和 max 表示选择选项的最小值和最大值。 其他输入值表示字符的长度
$.formValidator.retSetTipState	function(校验组号)表示把该组的提示内容恢复到 onshow 状态
$.formValidator.reloadAutoTip	重新定位自动构建的提示层

【例 6-20】 完成员工信息(姓名，工号，手机号码，E-mail)的添加功能，要求使用 formValidator 控件来校验用户输入数据的合法性。实现步骤如下。

(1) 创建一个名为 formValidatorTest 的文件夹，用于存放资源。

(2) 将下载的 formValidator 压缩包(本例采用 formValidator-4.0.1 版)解压，将 images 和 style 文件夹及下级所有文件复制到新建的 formValidator 文件夹下。

(3) 将 formValidator-4.0.1.min.js 和 formValidatorRegex.js 文件复制到 formValidator 文件夹下。

(4) 添加一个名为 formValidatorTest.html 的 HTML 文件。

(5) 打开 formValidatorTest.html 文件，添加对 jQuery 和 formValidator 文件的引用，代码如下：

```
<script src="../Scripts/jQuery.min.js"></script>
<link rel="stylesheet" type="text/css" href="formValidator/style/validator.
css">
<script src="formValidator/formValidator-4.0.1.min.js"></script>
<script src="formValidator/formValidatorRegex.js"></script>
```

(6) 添加校验控件的注册代码到<head></head>标签中，具体如下：

```
<script type="text/javascript">
    $().ready(function () {
        $.formValidator.initConfig({
            formID: "form1", debug: false, submitOnce: true,
            onError: function (msg, obj, errorlist) {
                $("#errorlist").empty();
                $.map(errorlist, function (msg) {
                    $("#errorlist").append("<li>"+msg+"</li>")
```

```
                }); 
                alert(msg);
            },
            submitAfterAjaxPrompt: '有数据正在异步验证,请稍等……'
        });
    });
</script>
```

(7) 添加信息录入表单,代码如下:

```
<form>
    <ul id="errorlist"></ul>
    <table>
        <tr>
            <td><span>姓名:</span></td>
            <td>
                <div><input name="name" id="name" /></div>
                <div id="nameTip" style="width:250px"></div>
            </td>
        </tr>
        <tr>
            <td><span>工号:</span></td>
            <td>
                <div><input name="number" id="number" /></div>
                <div id="numberTip" style="width:250px"></div>
            </td>
        </tr>
        <tr>
            <td><span>手机号码:</span></td>
            <td>
                <div><input name="cellphone" id="cellphone" /></div>
                <div id="cellphoneTip" style="width:250px"></div>
            </td>
        </tr>
        <tr>
            <td><span>Email:</span></td>
            <td>
                <div><input name="email" id="email" /></div>
                <div id="emailTip" style="width:250px"></div>
            </td>
        </tr>
        <tr>
            <td colspan="2">
                    <input type="submit" value="提交" />
            </td>
        </tr>
    </table>
</form>
```

(8) 为了使信息录入控件与消息提示框在同一行，需设置它们的 float 属性为 left，使其左向漂浮。在<head></header>标签中添加样式，代码如下：

```
<style type="text/css">
    div {
        float:left;
    }
</style>
```

(9) 在 $.formValidator.initConfig 代码后添加控件校验代码如下：

```
//配置检验姓名控件：不能为空，长度为 5~10 位
$("#name").formValidator({ onShow: "请输入用户名",
                           onFocus: "用户名至少为 5 个字符，最多为 10 个字符",
                           onCorrect: "用户名正确" })
           .inputValidator({
                           min: 5,
                           max: 10,
                           onError: "你输入的用户名非法，请确认" });
//配置校验工号控件：不为空，长度为 5 位
$("#number").formValidator({ onShow: "请输入工号，只有输入\"00186\"才是对的",
                             onFocus: "工号只能是 5 个字符",
                             onCorrect: "工号正确" })
             .inputValidator({ min: 5,
                             max: 5,
                             onError: "你输入的工号非法，请确认" });
//配置校验手机号码控件：使用 formValidatorRegex.js 文件中枚举的 mobile 正则表达式进
  行校验，长度必须为 11 位
$("#cellphone").formValidator({ empty: true,
                                onShow: "请输入你的手机号码，可以为空",
                                onFocus: "你要是进行了输入，必须输入正确",
                                onCorrect: "谢谢你的合作",
                                onEmpty: "你真的不想留手机号码吗?" })
                .inputValidator({min: 11,
                                max: 11,
                                onError: "手机号码必须是 11 位，请确认" })
                .regexValidator({ regExp: "mobile",
                                dataType: "enum",
                                onError: "你输入的手机号码格式不正确" });
//配置 E-mail 控件，长度为 6~100 位
$("#email").formValidator({ onShow: "请输入邮箱",
                          onFocus: "邮箱长度为 6~100 个字符",
                          onCorrect: "恭喜你输对了",
                          defaultValue: "@ @ " })
           .inputValidator({ min: 6,
                           max: 100,
                           onError: "邮箱有误，长度在 6~100 位，请确认!" })
           .regexValidator({
                   regExp: "^([\\w-.]+)@ @ (([[0-9]{1,3}.[0-9]{1,3}.[0-9]{1,
                           3}.)|(([\\w-]+.)+))"+" ([a-zA-Z]{2,4}|[0-9]{1,
                           3})(]?)$",
                   onError: "邮箱参考格式: michael@ 163.com" });
```

(10) 程序的运行界面如图 6-17 所示。不在控件中输入任何内容,直接单击"提交"按钮,运行界面如图 6-18 所示。

图 6-17 信息录入界面

图 6-18 表单验证结果

6.3.2 日期控件 My97DatePicker

My97DatePicker 是一款基于 jQuery 开发的日期选择控件,它支持静态限制、动态限制、脚本自定义限制,以及无效日期功能,利用这样功能可以任意定制不能选择的日期,这些日期即使毫无规律、毫无连续性,也可以通过这些功能的组合轻松搞定。它具有以下功能。

1. 更人性化、更全面的功能

大部分日期控件都具备以下功能,如带时间显示、支持周显示、自定义格式、自动纠错、智能纠错、起始日期、操作按钮自定义、快速选择日期、支持多种调用模式等,My97DatePicker 在这些方面做得更全面,更人性化,并且速度一流。

2. 强大的日期范围限制功能

可以自定义事件并有丰富的 API 库。

如果需要做一些附加的操作,日期控件自带的自定义事件可以满足需求。此外,还可以在自定义事件中调用系统提供的 API 库来做更多的运算和扩展,可以通过很少的代码满足个性化的需求。

3. 多语言支持和自定义皮肤支持

通过 lang 属性,可以为每个日期控件单独配置语言,当然也可以通过 WdatePicker.js 配置全局的语言,皮肤也一样,只要配置 skin 属性即可。这样一个页面中可以显示多种语言、多种皮肤的日期控件,并且它们之间的切换可以不刷新页面。

4. 跨无限级框架显示和自动选择显示位置

无论把日期控件放在哪里,都不需要担心会被外层的 iframe 所遮挡进而影响客户的体验,My97 日期控件是可以跨无限级框架显示的,并且当控件处在页面边界时,它会自动选择显示的位置。此外,还可以使用 position 参数对弹出位置做调整。

My97DatePicker 插件及帮助文档的下载地址为 http://www.my97.net/，例 6-21 将演示如何使用 My97DatePicker 插件。

【例 6-21】 单击文本框时弹出日期控件，使用户可以从日期控件中选择日期。

实现步骤如下：

(1) 创建一个名为 DatePickerTest 的文件夹，用于存放资源。

(2) 将解压后的 My97DatePicker 文本夹复制 DatePickerTest 文件夹下。

(3) 新建一个名为 datePickTest.html 的 HTML 文件，在＜head＞＜/head＞标签中引用 JQuery 文件及日期控件，代码如下：

```
<script type="text/javascript" src="../Scripts/jQuery.min.js"></script>
<script type="text/javascript" src="DatePickTest/My97DatePicker/WdatePicker.js">
</script>
```

(4) 在 datePickTest.html 文件的＜body＞＜/body＞标签中添加一个文本框，代码如下：

```
<span>出生日期：</span><input id="txtBirthDay" onclick="WdatePicker()" />
```

(5) 运行 datePickTest.html 文件，界面如图 6-19 所示。



图 6-19　日期插件程序的运行结果

6.4　综合实例

6.4.1　需求描述

创建 HTML 页面，实现员工信息的录入功能，员工信息包括姓名、性别、身份证号码、基本工资和工作经历信息（时间段、公司名称、职务、主要工作及职责、证明人）。

综合运用 jQuery 的相关知识实现如下需求。

(1) 时间段应引用 jQuery 日期插件。

(2) 可以实现同时录入多条工作经历信息。

(3) 可以对工作经历信息进行移除。

6.4.2 分析及实现

整个页面分为 2 块区域，员工基本信息录入区域和工作经历录入区域。员工基本信息录入区域主要用于放置员工基本信息的录入控件，即放置姓名、性别、身份证和基本工资等信息的录入控件；工作经历录入区域则用于放置工作经历的相关信息录入控件，即放置时间段、公司名称、职务、主要工作及职责、证明人等信息的录入控制。将界面布局设置如图 6-20 所示。

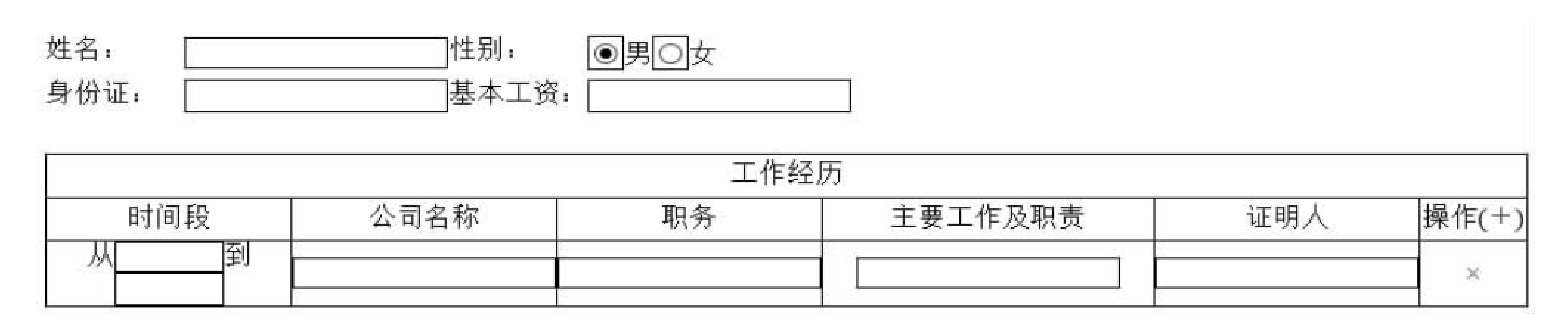


图 6-20 员工信息录入界面

图 6-20 中操作列的“＋”可以动态地添加行，为增加新的工作经历信息添加控件；“×”可以删除所在的行，但不能删除第 1 行。

实现步骤如下：

(1) 新建一个“jQuery 综合实例”的文件夹用于存放资源。

(2) 新建一个名为“jQuery 综合实例.htm”的 HTML 页面。

(3) 打开“jQuery 综合实例.html”页面，在<head></head>中引用 jQuery 库文件。

```
<script src="../Scripts/jQuery.min.js"></script>
```

(4) 在<head></head>中添加学生基本信息录入控件的样式表如下：

```
<style type="text/css">
    /* 设置 input 标签的样式 */
    input{
        border: 1px solid black;     /* 将边框的宽度设置为 1px、黑色实线 */
    }

    /* 员工基本信息标准样式 */
    .title {
        width: 80px;
    }

    /* 员工信息表单样式 */
    .rowInfo {
        margin-bottom: 5px;
        clear: both;                 /* 清除前面控件的左漂浮效果 */
    }
</style>
```

(5) 在<body></body>中添加学生基本信息录入表单。

```
<div class="rowInfo">
    <div class="title">
        <span>姓名:</span></div><div><input id="name" name="name" />
    </div>
    <div class="title">
        <span>性别:</span></div><div>
        <input type="radio" checked="checked" name="gender">男</input>
        <input type="radio" name="gender">女</input>
    </div>
</div>
<div class="rowInfo">
    <div class="title"><span>身份证:</span></div>
    <div><input id="id" name="id" /></div>
    <div class="title"><span>基本工资:</span></div>
    <div><input id="salary" name="salary" /></div>
</div>
```

(6) 运行界面如图 6-21 所示。



图 6-21　学生基本信息录入界面

(7) 添加工作经历录入控件样式。

```
/* 设置超联结的样式 */
a {
    text-decoration:none;              /* 去掉下划线 */
}

/* 设置通用单元格样式 */
.tbl {
    border-top: 1px solid black;       /* 将上边框的宽度设置为 1px、黑色实线 */
    border-left: 1px solid black;      /* 将左边框的宽度设置为 1px、黑色实线 */
    clear: both;                       /* 清除其他控件的漂浮效果 */
    margin-top: 50px;                  /* 设置下边界为 50 像素 */
}

/* 设置工作经历表中单元格的样式 */
.tbl td {
    border-right: 1px solid black;
    border-bottom: 1px solid black;
    text-align:center;                 /* 内容居中 */
    height:25px;
```

```
}

/ * 日期录入框的样式 * /
.date {
    width:59px;
}
```

(8) 按需求,用户可以录入多条工作经历,类似的多信息录入采用 table 标签进行布局,可以大大地简化实现的难度,在<body></body>标签内添加如下 HTML 代码。

```
<table border="0" id="tblWorkExperience" class="tbl" cellpadding="0"
cellspacing="0">
    <thead><tr><td colspan="6">工作经历</td></tr></thead>
    <tr>
        <td width="160px">时间段</td><td width="130px">公司名称</td>
        <td width="100px">职务</td><td width="200px">主要工作及职责</td>
        <td width="100px">证明人</td>
        <td width="70px">操作 (<a href="#" onclick="addRow()">+</a>)</td>
    </tr>
    <tr id="r_1">
        <td>从<input id="txtFromDate_1"  onclick="WdatePicker()"
            class="date" />到<input id="txtToDate_1" class="date"
            onclick="WdatePicker()"  />
        </td>
        <td><input id="txtCompanyName_1" /></td>
        <td><input id="txtPostion_1" /></td>
        <td><input id="txtResponsibility_1" /></td>
        <td><input id="txtWitness_1"></td>
        <td><a href="#" onclick="return false;" style=" color:gray;" title=
        "删除">×</a></td>
    </tr>
</table>
```

上述代码中的"_1"作为行序号的标识,代表第 1 行;代码块 onclick="WdatePicker()"将为控件注入日期选择功能。

(9) 程序的运行结果如图 6-22 所示。

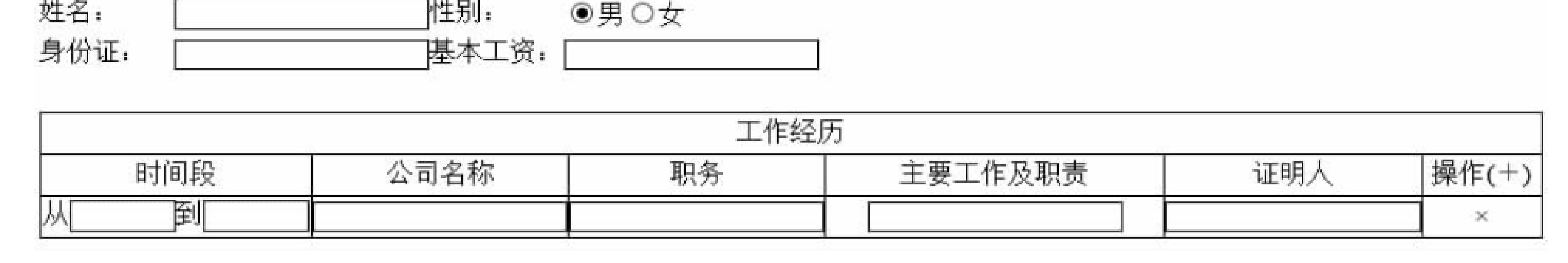


图 6-22　员工信息详细录入界面

(10) 当用户单击图 6-22 中的"+"时,页面将自动在工作经历列表中增加一行工作经历录入控件,使用 jQuery 的 append 方法可向<table></table>标签中增加行,代码如下:

```
<script type="text/javascript">
    //定义工作经历表的行计数器,用于标识行序列号,并作为 id 的组成部分,其值只增不减
    var rowCounter=1;
    //定义增加行函数
    function addRow(){
        //行号加 1
        rowCounter++;
        //使用 ID 选择器找到 tblWorkExperience 标签,并使用 append 方法添加新行
        $("#tblWorkExperience").append('<tr id="r_'+rowCounter+'">
            <td>从<input onclick="WdatePicker()" id="txtFromDate_'+
             rowCounter+'" class="date" />到<input onclick="WdatePicker()"
             id="txtToDate_' +rowCounter+'" class="date" /></td><td><input id
             ="txtCompanyName_'+rowCounter +'"/>
            </td><td><input id="txtPostion_'+
             rowCounter+'"/></td><td><input id="txtResponsibility_'+
             rowCounter+'"/></td><td><input id="txtWitness_'+
             rowCounter+'"></td><td><a href="#" onclick="removeRow('+
             rowCounter+')" title="删除">×</a></td>'+
             '</tr>');
   }
</script>
```

(11) 上述代码已实现添加工作经历行的功能。当然也可以移除行中的控件,在 addRow 方法后添加添加 removeRow 方法,使其具备根据行标识移除行标签的功能,代码如下:

```
//根据行序号移除工作经历表中的行
function removeRow(rowIndex) {
    if(confirm("真的打算移除吗?")) {
        $("#r_"+rowIndex).remove();
    }
}
```

(12) 添加日期控件。基于 B/S 的日期控件需要自定义控件,jQuery 联盟提供了大量开源、免费的日期控件,可以从 http://JQueryui. com/下载得到。本书采用的 My97DatePicker 日期控件可以从 http://www. my97. net/下载。

(13) 将下载的整个 DatePicker 包复制到“jQuery 综合实例”文件夹中,再在“jQuery 综合实例. html”文件的 jQuery 引用后添加对其文件的引用,代码如下:

```
<script src="My97DatePicker/WdatePicker.js"></script>
```

(14) 运行界面如图 6-23 所示。



图 6-23　jQuery 综合实例程序的运行结果

本章小结

jQuery 是基于 JavaScript 编制而成的应用程序接口（application programming interface，API），开发者可以以很少的代码方便、快捷地调用这些应用程序接口完成具体功能，大大地简化了 JavaScript 的应用复杂性。jQuery 是一个开源的 JavaScript API，它不但提供了大量基本的应用接口，还允许开发者对其进行扩展，经过多年的发展，jQuery 拥有大量成熟的第三方控件，可以大大地减少开发的工作量，节省软件开发成本。

本章详细讲解了 jQuery 在开发过程中用到的实用功能，涵盖了选择器、事件方法、动画、DOM 操作等知识点。本章以“实用、够用”作为基本编写思路，案例简洁易懂又实用，讲解过程中辅以开发场景，介绍了使用技巧，并详细阐述了分析及设计过程，可以很好地帮助学习者理解及掌握。

第 7 章　客户端数据请求技术

7.1　客户端请求技术简介

动态网页并不是指在页面上具有动画功能的网页，而是指网页内容可根据不同的业务情况动态变更显示内容的网页，一般情况下动态网页除了要设计网页外，还要通过编写业务处理程序和数据库来使网页具有更多自动的和高级的功能。即使一个网页中没有动画效果，但是实现了前后台交互功能的也叫作动态网页。程序是否在服务器端运行，是动态网页的重要标志。在服务器端运行的程序、网页、组件属于动态网页，它们会随不同客户、不同时间而返回不同的网页，例如 ASP、PHP、JSP、CGI 等。运行于客户端的程序、网页、插件、组件属于静态网页，例如 HTML 页、Flash、JavaScript、VBScript 等，它们是永远不变的。

在最开始的动态网站开发中用到的 PHP、ASP、JSP，都是将 HTML 代码和后台代码嵌套在同一个页面之中形成服务器端动态页面。对于业务不太复杂的网站来说，这种开发方式简单明快。然而业务总会变得复杂，这种将前后台代码嵌套在一起的方式也暴露出它最大的弊端就是代码的可读性和可维护性变得越来越差，往往带来维护成本的大量增加。这个时候如何让前后端更合理地进行分工，如何提高代码的可维护性，在 Web 开发中变得更重要了。

这之后越来越多的 Web 开发环境开始支持 MVC(model view contorller)的设计模型，为开发者提供了全套的开发框架，也出现了各种各样的 Web 开发框架。前端页面主要负责显示，业务逻辑运行在服务器端，使得前后端分工更清晰了。这一技术变革促进了 Web 网站的大发展，大量的网站被开发出来。如何重用其他已经开发出来的网站的功能，构成一个强大的分布式应用系统，在 Web Service 出现之后也变成了现实。

将前后端分开之后，它们之间的通信就只能通过 HTTP 请求来完成。但是这样的程序在用户每次和后台交互的过程中，都要等待服务器的返回才能继续往下操作，用户体验感不如桌面程序好。而且大量的业务程序运行在服务器端，对于服务器和网络带宽都是不小的负担。

ajax 的提出，让人们看到了网页在不等到服务器返回的时候，也可以继续往下操作，网页也能做出和桌面程序一样的体验效果，这让人们将目光重新汇聚到了前端开发技术上。这时大量的前端框架开始出现，如 jQuery 等。在这些框架中不但封装了前端页面的显示效果，也封装了服务器的交互技术。

7.2　Web Service 简介

在日常的系统开发中可能会遇到一些系统设计之外的需求，比如在开发某旅游平台的时候，用户需要查询目的地的天气情况来安排自己的行程和规划自己的随身衣物等，这样的设计使系统更加人性化，更好地满足用户的需求。

首先天气预报的数据是实时变化的，需要实时收集，这不是旅游平台的主要功能；其次目的地数量庞大，天气预报的数据量势必也会很庞大。所以在设计时也不能把这部分数据存入数据库中，程序员也没有必要为了天气预报开发这么多代码，就算真的开发出来，其工作量可能大于旅游平台的开发。但是天气预报的功能在旅游平台中又是一个不可或缺的功能。

如果有一种解决方案能实现传入一个目的地名称，就能返回对应城市的天气预报的实时数据，将为旅游平台的开发提供极大的便利，而且其访问的形式就像访问某个天气预报网站一样便利，还不受开发语言的限制，提供的天气预报的返回结果可以在不同的开发语言中解析出相同的天气预报结果，这就是 Web Service 所能实现的功能。

Web Service 是一种可以接收从互联网上的其他系统中传递过来的请求，它是一种轻量级的独立的通信技术，它使运行在不同机器上的不同应用程序无须借助附加的、专门的第三方软件或硬件，就可相互交换数据或集成，所以它是一个平台独立的、低耦合的、自包含的、基于可编程的 Web 的应用程序，适用于开发分布式的互操作的应用程序，减少了应用程序接口的花费，也为各个不同系统的集成提供了一个通用机制。Web Service 基于一些常规的产业标准以及已有的一些技术，它使用开放的 XML 标准来描述、发布、发现、协调和配置这些应用程序，通过 SOAP 在 Web 上提供软件服务，使用 WSDL 文件进行说明，并通过 UDDI 进行注册。

XML(extensible markup language，扩展型可标记语言)面向短期的临时数据处理、面向万维网络，是 SOAP 的基础。

SOAP(simple object access protocol，简单对象存取协议)是 XML Web Service 的通信协议。当用户通过 UDDI 找到某个在线服务的 WSDL 描述文档后，就可以通过 SOAP 来调用该 Web 服务中的一个或多个操作。SOAP 是 XML 文档形式的调用方法的规范，它可以支持不同的底层接口，如 HTTP(S)或者 SMTP。

WSDL(web services description language，网络服务描述语言)文件是一个 XML 文档，用于说明一组 SOAP 消息以及如何交换这些消息。大多数情况下由软件自动生成和使用。

UDDI(universal description，discovery and integration，通用描述、发现与集成)是一个主要针对 Web 服务供应商和使用者的新项目。在用户能够调用 Web 服务之前，必须确定这个服务内包含哪些商务方法，找到被调用的接口定义，还要在服务端来编制软件。UDDI 是一种根据描述文档来引导系统查找相应服务的机制。UDDI 利用 SOAP 消息机制(标准的 XML/HTTP)来发布、编辑、浏览以及查找注册信息。它采用 XML 格式来封装各种不同类型的数据，并且发送到注册中心或者由注册中心来返回需要的数据。

7.2.1 XML 文件

XML 是用于标记电子文件,使其具有结构性的标记语言。它可以用来标记数据、定义数据类型,是一种允许用户对自己的标记语言进行定义的源语言。它是标准通用标记语言(SGML)的子集,非常适合 Web 传输。它提供统一的方法来描述和交换独立于应用程序或供应商的结构化数据。是 Internet 环境中跨平台的、依赖于内容的技术,也是当今处理分布式结构信息的有效工具。早在 1998 年 2 月,W3C 就发布了 XML 1.0 规范,使用它来简化因特网的文档信息传输。简单地说,XML 就是一种平台无关的数据的描述语言,因此,可以在完全不同的系统之间实现共享数据。虽然它是语言,但是通常情况下,它并不具备常见语言(如 HTML)的基本功能——被计算机识别并运行。只有依靠另一种语言解释它,才能使它达到你想要的效果或被计算机所接受。例 7-1 给出了一个描述天气预报的 XML 文件。

【例 7-1】 天气预报的 XML 文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<cities>
    <city id="010">
        <name>北京</name>
            <describe>北京位于华北平原北部,背靠燕山,毗邻天津市和河北省。北京的气候
            为典型的北温带半湿润大陆性季风气候。</describe>
            <weather id="today">
                <date>2016 年 11 月 2 日</date>
                <situation>多云</situation>
                <temperature>0℃/7℃</temperature>
            </weather>
            <weather id="tomorrow">
                <date>2016 年 11 月 3 日</date>
                <situation>多云转晴</situation>
                <temperature>1℃/10℃</temperature>
            </weather>
    </city>
    <city id="023">
            <name>重庆</name>
            <describe txt="重庆,简称巴和渝,别称巴渝、山城、渝都、桥都、雾都,
                是中华人民共和国中央直辖市、国家中心城市、超大城市。"/>
        <weather id="today">
            <date>2016 年 11 月 2 日</date>
            <situation>晴</situation>
            <temperature>15℃/25℃</temperature>
        </weather>
        <weather id="tomorrow">
            <date>2016 年 11 月 3 日</date>
            <situation>晴转阴</situation>
            <temperature>15℃/23℃</temperature>
        </weather>
    </city>
</cities>
```

从上面的 XML 清单中可以看出 XML 文件的一些重要的特征。

1. XML 声明

与定义 HTML 文件等其他标注性语言一样，在定义 XML 文件时，也要进行声明。例 7-1 中的第 1 行给出了 XML 文件的定义如下：

```
<?xml version="1.0" encoding="UTF-8"?>
```

其中，xml 是 XML 的关键字，是 XML 语言中保留的，且必须是小写。另外第一个问号和 xml 关键字之间不能有空格。version 为 XML 文档的版本，它是由万维网联盟（W3C）发布的。encoding 为文档编码类型，XML 文件默认采用 UTF-8 的字符编码方案。

2. XML 元素

在第 2 章中我们讲到在 HTML 语言中预先定义了如<a>、<hr>等标签，它们称为 HTML 的元素，且每个标签都有特殊的含义和功能。与 HTML 类似，XML 文件的内容也是定义在标签之内的，它们就称为 XML 的元素。如例 7-1 中的第 2 行开始就定义了很多的 XML 元素。如 cities、city、name 等。在 XML 文档中每一个元素都必须要有对应的结束标记才被认为是格式正确的。在开始标记和结束标记之间的内容称为元素的值。如例 7-1 中的第 4 行定义的 name 元素的值为北京，如下所示。

```
<name>北京</name>
```

元素值的部分也可以换为 XML 元素，该元素就是子元素，这就实现了元素的嵌套。这点和 HTML 的元素嵌套也是类似的。如例 7-1 中的第 7～11 行，如下所示。

```
<weather id="today">
    <date>2016 年 11 月 2 日</date>
    <situation>多云</situation>
    <temperature>0℃/7℃</temperature>
</weather>
```

weather 元素就是 date、situation、temperature 元素的父元素。当然整体来看例 7-1 也是一个大的嵌套结构，读者可自行去分析其中的元素嵌套过程。

不同于 HTML 语言，在 XML 文件中允许使用者按照自己的意愿自由地为 XML 元素命名。W3C XML 1.0 规范规定元素必须以下面这些字符打头：从 A 到 Z，或者从 a 到 z，或者以“_”打头。也就是说大小写的英文字符或者下划线。如在例 7-1 中定义的各种元素的名称。

3. XML 属性

在 HTML 中可以为元素定义属性，用于帮助描述元素，如超链接标签“<a>”需要定义“href”属性才能实现超链接访问。同样的，在 XML 文件中也可以为元素定义属性，如例 7-1中的第 3 行和第 7 行都为 city 元素定义了 id 属性。当然也可以为一个元素定义多个的属性，甚至可以将子元素的值也定义为属性，用来减少标签的数量，如下所示。

```
<city id="023" name="重庆">
```

正是因为有了属性,才可以在一个标签中包含两个以上标签相同但属性不同的元素,这点类似于在数据库中主键的值不能重复一样。如例 7-1 中,cities 元素中包含了两个 city 元素,在一个 city 元素中包含了两个 weather 元素。其中属性的命名也有严格的规则,必须以从 A 到 Z,或者从 a 到 z 打头。也就是说以大小写的英文字符打头。

4. XML 的重要原则

为了保证 XMl 文件在不同的解释器和应用程序之间正确地工作,XML 文件必须严格地遵守几条基本的 XML 原则。这些原则如下:

(1) 一个 XML 文件只包含一个根元素。

例 7-1 中给出的 XML 文档中可以清晰地识别出文档的根元素为 cities。根元素是 XML 解析器解析 XML 文件的一个引用点,一个 XML 中有且仅有一个根元素。如果将一个 XML 文件看成一棵树,根元素就是这棵树的根节点,而其他的子元素就是这棵树的叶子节点。

(2) XML 元素必须有个结束标记。

在 HTML 文件中类似于"<p>hello word"这样没有关闭的标记可以被浏览器正确地解释执行,但是在 XML 文件中这是不允许的。如果某个标记没有被关闭,解析器会报出找不到结束标记的错误。在 XML 文件中有两种关闭标记的方法。

第一种是用"/"加上元素名进行关闭,如例 7-1 中第 4 行的</name>。

第二种是对于那种没有包含数据或者只含有属性的元素可以自关闭,如例 7-1 中第 20 行的<describe/>,这种方式比<describe></describe>更简洁。

(3) XML 元素的属性都必须用引号括起来。

对于例 7-1 中的第 3 行 ctiy 元素的 id 属性,必须要用引号扩展起来,这样解析器才能正确地解析 XML 文件。

(4) XML 元素可以包含子元素,但是子元素必须正确地嵌套。

例 7-1 的第 3～17 行和第 18～31 行中都给出了北京和重庆的天气预报信息,它们的元素内容是一样的,从中也可以看出元素是可以嵌套子元素的。在城市中嵌套了城市的名称和描述,以及今明两天的天气情况。在天气情况的描述也往下可以嵌套多个层次。但是注意在一层嵌套没有完成之前不要引入下一层的元素。

(5) XML 元素对于大小写是敏感的。

在 HTML 中,浏览器对标签的大小写不敏感。但是在 XML 文件中,如果将例 7-1 的第 4 行改为"<name>北京<Name>",XML 文件解析器会产生一条"结束标签 Name 不匹配开始标签 name"的错误。

7.2.2 Web Service 原理

Web Service 的三大要素为 SOAP、WSDL 和 UDDI。其中 SOAP 用来描述传递信息的格式,WSDL 用来描述如何访问具体的接口,UDDI 用来管理、分发、查询。简单地讲,Web 服务是一个 URL 资源,客户端可以通过编程方式请求得到它的服务,而不需要知道所请求的服务是怎样实现的,这一点与传统的分布式组件对象模型是不相同的。Web 服务的体系结构是基于 Web 服务提供者、Web 服务请求者、Web 服务注册中心三个角色和发布、发现、绑定三个动作构建的。也就是说 Web 服务提供者就是 Web 服务的拥有者,它耐心等待为

其他服务和用户提供自己已有的功能；Web 服务请求者就是 Web 服务功能的使用者，它利用 SOAP 消息向 Web 服务提供者发送请求以获得服务；Web 服务中介者的作用是把一个 Web 服务请求者与合适的 Web 服务提供者联系在一起，它充当管理者的角色，一般是 UDDI。这三个角色是根据逻辑关系划分的，在实际应用中，角色之间很可能有交叉：一个 Web 服务既可以是 Web 服务提供者，也可以是 Web 服务请求者，或者两者兼而有之。图 7-1 显示了 Web 服务角色之间的关系。

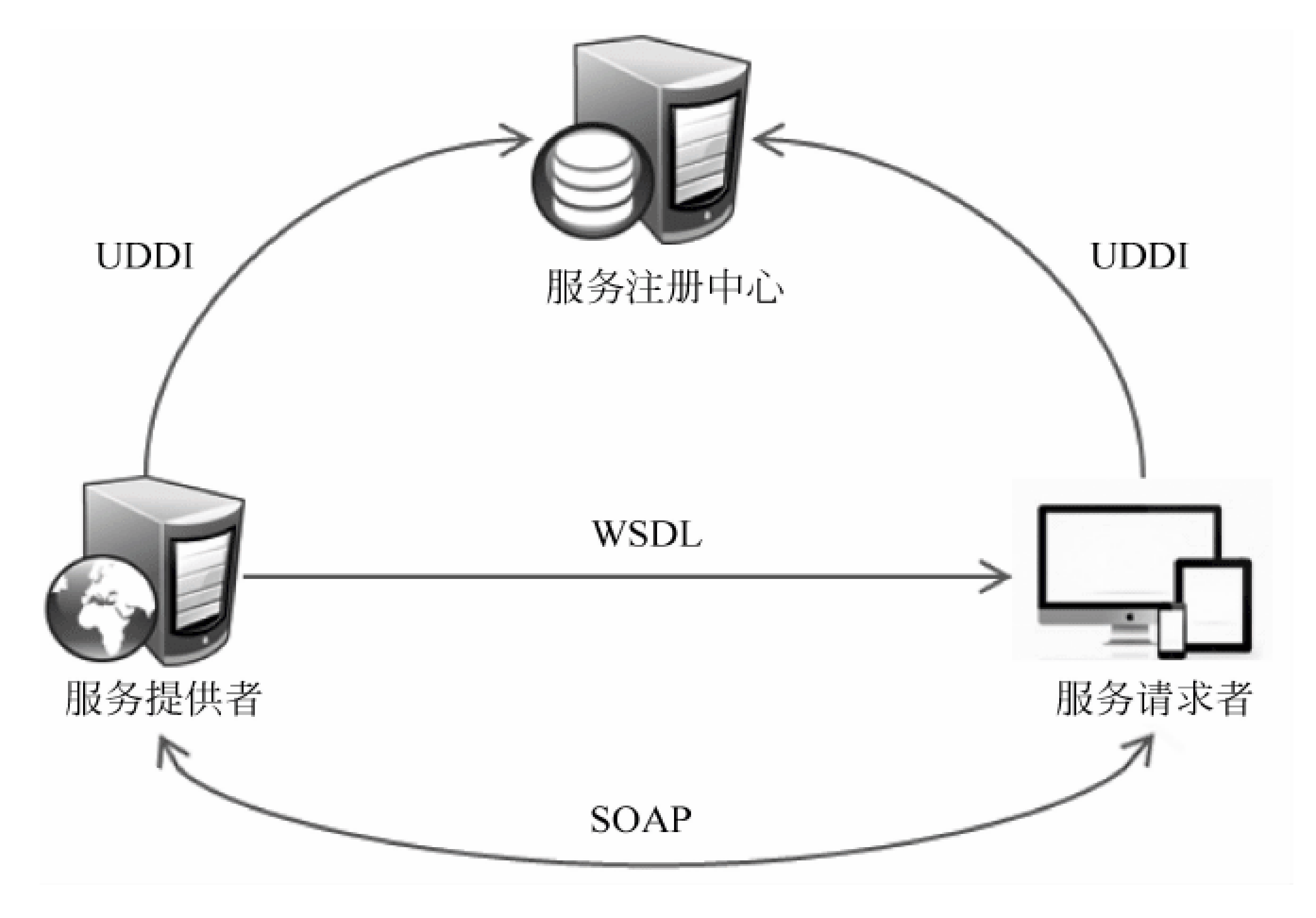


图 7-1　Web Service 架构流程图

由图 7-1 可以看出，实现一个完整的 Web 服务包括以下步骤。

(1) Web 服务提供者设计实现 Web 服务，将调试正确后的 Web 服务通过 Web 服务中介者发布，并在 UDDI 注册中心注册，这个过程也称为发布的过程。

(2) Web 服务请求者向 Web 服务注册中心请求特定的服务，注册中心根据请求查询 UDDI 注册中心，为请求者寻找满足请求的服务。

(3) Web 服务注册中心向 Web 服务请求者返回满足条件的 Web 服务描述信息，该描述信息用 WSDL 写成，各种支持 Web 服务的客户端都能阅读。

(4) 利用从 Web 服务中介者返回的描述信息生成相应的 SOAP 消息，发送给 Web 服务提供者，以实现 Web 服务的调用。

(5) Web 服务提供者按 SOAP 消息执行相应的 Web 服务，并将服务结果返回给 Web 服务请求者。

1. WSDL 文档

WSDL 也就是 Web 服务描述语言，是基于 XML 的用于描述 Web 服务以及如何访问 Web 服务的语言。它是一种使用 XML 编写的文档。这种文档可描述某个 Web Service。它可规定服务的位置，以及此服务提供的操作(或方法)。简单地说，它仅仅是一个简单的 XML 文档。一个 WSDL 文档通常包含有以下元素，即 types、message、portType、operation、binding、service、port 元素。这些元素嵌套在 definitions 元素中，该元素封装了整个文档，同时通过其 targetNamespace 命名空间提供了一个 WSDL 文档。除了提供一个命名空间外，该元素没有其他作用，故不作详细描述。详细的结构如下所示。

```
<?xml version='1.0' encoding='UTF8'?>
<wsdl:definitions>
    <wsdl:types></wsdl:types>
    <wsdl:message></wsdl:message>
    <wsdl:portType>
        <wsdl: operation></wsdl: operation>
    </wsdl:portType>
    <wsdl:binding></wsdl:binding>
    <wsdl:service></wsdl:service>
<wsdl:definitions>
```

(1) types：WSDL 采用了 W3C XML 模式内置类型作为其基本类型系统。它用作一个容器，用于定义 XML 模式内置类型中没有描述的各种数据类型。如下是中国气象局天气预报 Web Service 接口（http://www.webxml.com.cn/WebServices/WeatherWebService.asmx?wsdl，该站点的数据 2.5 小时左右自动更新一次，包括 340 多个中国主要城市和 60 多个国外主要城市三日内的天气预报数据）的取得支持城市的 type 定义，具体定义如下所示。

```
<wsdl:types>
    <s:schema targetNamespace="http://WebXml.com.cn/" elementFormDefault="
    qualified">
        <s:element name="getSupportCity">
            <s:complexType>
                <s:sequence>
                    <s:element name="byProvinceName" type="s:string"
                    maxOccurs="1" minOccurs="0"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="getSupportCityResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="getSupportCityResult"
                    type="tns:ArrayOfString" maxOccurs="1" minOccurs="0"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:complexType name="ArrayOfString">
            <s:sequence>
                <s:element name="string" type="s:string"
                maxOccurs="unbounded" minOccurs="0" nillable="true"/>
            </s:sequence>
        </s:complexType>
    </s:schema>
</wsdl:types>
```

此段定义描述了 getSupportCity 方法的传入参数的参数名称为 byProvinceName 且类型为字符串类型；它的响应信息为 getSupportCityResponse，名称为 getSupportCityResult

且类型为自定义的 ArrayOfString 类型。定义的最后对 ArrayOfString 进行了详细的说明。

(2) message：描述通信消息的数据结构的抽象类型化定义，使用 types 描述的类型来定义输出或者接受消息的数据结构，它的定义如下所示。

```
<wsdl:message name="getSupportCitySoapIn">
    <wsdl:part name="parameters" element="tns:getSupportCity"/>
</wsdl:message>
<wsdl:message name="getSupportCitySoapOut">
    <wsdl:part name="parameters" element="tns:getSupportCityResponse"/>
</wsdl:message>
```

(3) portType 和 operation：描述服务和服务的方法。operation 的输入和输出直接引用 message 的描述。也就是说它定义了 Web 服务的抽象接口。该接口类似编程语言中的接口，都是只定义了一个抽象方法，没有定义实现。一个 portType 中可以定义多个 operation，一个 operation 可以看作一个方法，它的定义如下所示。

```
<wsdl:portType name="WeatherWebServiceSoap">
    <wsdl:operation name="getSupportCity">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
        <br /><h3>查询本天气预报 Web Services 支持的国内外城市或地区信息</h3>
        <p>输入参数：byProvinceName=指定的洲或国内的省份，若为 ALL 或空则表示返回全
        部城市。返回数据：一个一维字符串数组 String()；结构为：城市名称(城市代码)。<
        /p><br /></wsdl:documentation>
        <wsdl:input message="tns:getSupportCitySoapIn"/>
        <wsdl:output message="tns:getSupportCitySoapOut"/>
    </wsdl:operation>
</wsdl:portType>
```

(4) binding：描述 Web Services 的通信协议，它将一个抽象 portType 映射到一组具体协议(SOAP 和 HTTP)、消息传递样式、编码样式中，通常与协议专有的元素和在一起使用。它还要描述 Web Services 的方法以及输入和输出，它的定义如下所示。

```
<wsdl:binding name="WeatherWebServiceSoap" type="tns:WeatherWebServiceSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="getSupportCity">
        <soap:operation style="document" soapAction="http://WebXml.com.cn/
        getSupportCity"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
```

上面的定义中将服务访问点与 SOAP/HTTP 协议进行了绑定，且定义了在具体 SOAP 调用时应当使用的 soapAction 是 getSupportCity 函数。

(5) service 和 port：描述 Web Services 访问点的集合。一个 service 元素可以包含一个或者多个 port 元素，其中每个 port 元素表示一个不同的 Web 服务。port 元素将 URL 赋给一个特定的 binding，甚至可以使两个或者多个 port 元素将不同的 URL 赋值给相同的 binding。因为包括 SOAP 1.1 和 SOAP 1.2 的描述，所以一个方法有对应的两种描述，如下所示。

```
<wsdl:service name="WeatherWebService">
    <wsdl:port name="WeatherWebServiceSoap" binding="tns:WeatherWebServiceSoap">
        <soap:address location="http://www.webxml.com.cn/
                      WebServices/WeatherWebService.asmx"/>
    </wsdl:port>
    <wsdl:port name="WeatherWebServiceSoap12"
    binding="tns:WeatherWebServiceSoap12">
        <soap12:address location="http://www.webxml.com.cn/
                        WebServices/WeatherWebService.asmx"/>
</wsdl:port>
```

WSDL 文档是在发布 Web Service 程序的时候自动生成的。在日常使用中要会阅读它，这样在调用时才能知道如何正确地去使用已经提供的服务。

2. SOAP 协议

SOAP 也就是简单对象访问协议，是用于应用程序之间进行通信的一种通信协议，是一种轻量的、简单的、基于 XML 的协议，它被设计成在 Web 上交换结构化的和固化的信息。它是一种用于访问 Web 服务的协议。SOAP 基于 XML 和 HTTP，通过 XML 来实现消息描述，然后再通过 HTTP 实现消息传输。正是因为它基于 XML 和 HTTP，所以其独立于语言、独立于平台，并且因为 XML 的扩展性很好，所以基于 XML 的 SOAP 扩展性也很好。

3. UDDI

UDDI 是统一描述、发现和集成的缩写。它是一个基于 XML 的跨平台的描述规范，可以使世界范围内的企业在互联网上发布自己所提供的服务。它使企业在互联网上可以互相发现并且定义业务之间的交互。它是核心的 Web 服务标准之一。它通过 SOAP 进行消息传输，用 WSDL 描述 Web 服务及其接口的使用，主要用来注册和查找服务，提供一种发布和查找服务描述的方法。

7.2.3 Web Service 的调用

现在有很多的 Web Service 已经被开发出来，供大家使用，比如在 7.2.1 小节中提到的天气预报的 Web Service。本节用 Java 语言和 C# 语言来分别说明在 Web 服务器端程序中如何调用这个天气预报服务。

1. 在 Java 中调用 Web Service

【例 7-2】 Java 中调用 Web Service。

(1) 下载相应的 Axis2 插件的相关 Jar 包(见图 7-2)。

下载地址为：http://axis.apache.org/axis2/java/core/download.html。

AXIS2/JAVA
Home
Downloads
Release Notes
1.6.1
1.6.2
1.6.3
1.6.4
1.7.0
1.7.1
1.7.2
1.7.3
1.7.4
Modules

Releases

The current release is 1.7.4 and was published on October 21, 2016. The release note for this release can be found here.

The following distributions are available for download:

	Link	Checksums and signatures
Binary distribution	axis2-1.7.4-bin.zip	MD5 SHA1 PGP
Source distribution	axis2-1.7.4-src.zip	MD5 SHA1 PGP
WAR distribution	axis2-1.7.4-war.zip	MD5 SHA1 PGP
Service Archive plugin for Eclipse	axis2-eclipse-service-plugin-1.7.4.zip	MD5 SHA1 PGP
Code Generator plugin for Eclipse	axis2-eclipse-codegen-plugin-1.7.4.zip	MD5 SHA1 PGP
Axis2 plugin for IntelliJ IDEA	axis2-idea-plugin-1.7.4.zip	MD5 SHA1 PGP

图 7-2　Axis 插件的下载

下载 WAR distribution 和 Code Generator plugin for Eclipse 两个包。其中 WAR distribution 中包含的是 Java Web Service 的支持类的包文件。Code Generator plugin for Eclipse 是 Eclipse 的 Web Service 的插件。

（2）安装 Eclipse 的 Web Service 插件。

解压 Code Generator plugin for Eclipse 包，将解压后 plugins 文件夹的文件复制到 Eclipse 安装目录的 plugins 文件夹下。重启 Eclipse，在 File→New→Other...命令打开的对话框中能找到 Axis2 Wizards 目录，如图 7-3 所示。

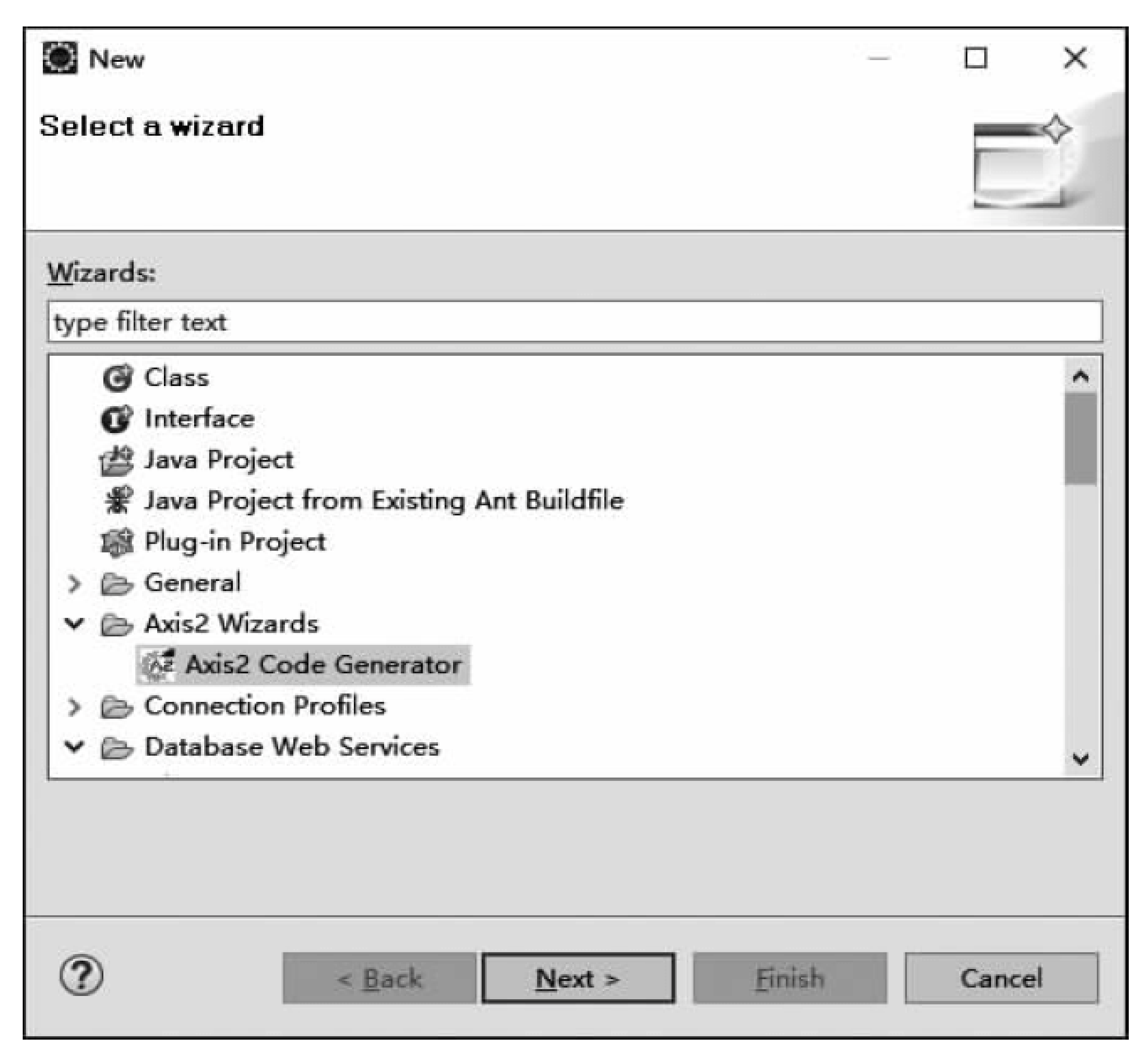


图 7-3　Web Service 插件

（3）生成 Web Service 客户端代码。

单击图 7-3 中的 Next 按钮，进入代码生成界面，如图 7-4 所示。

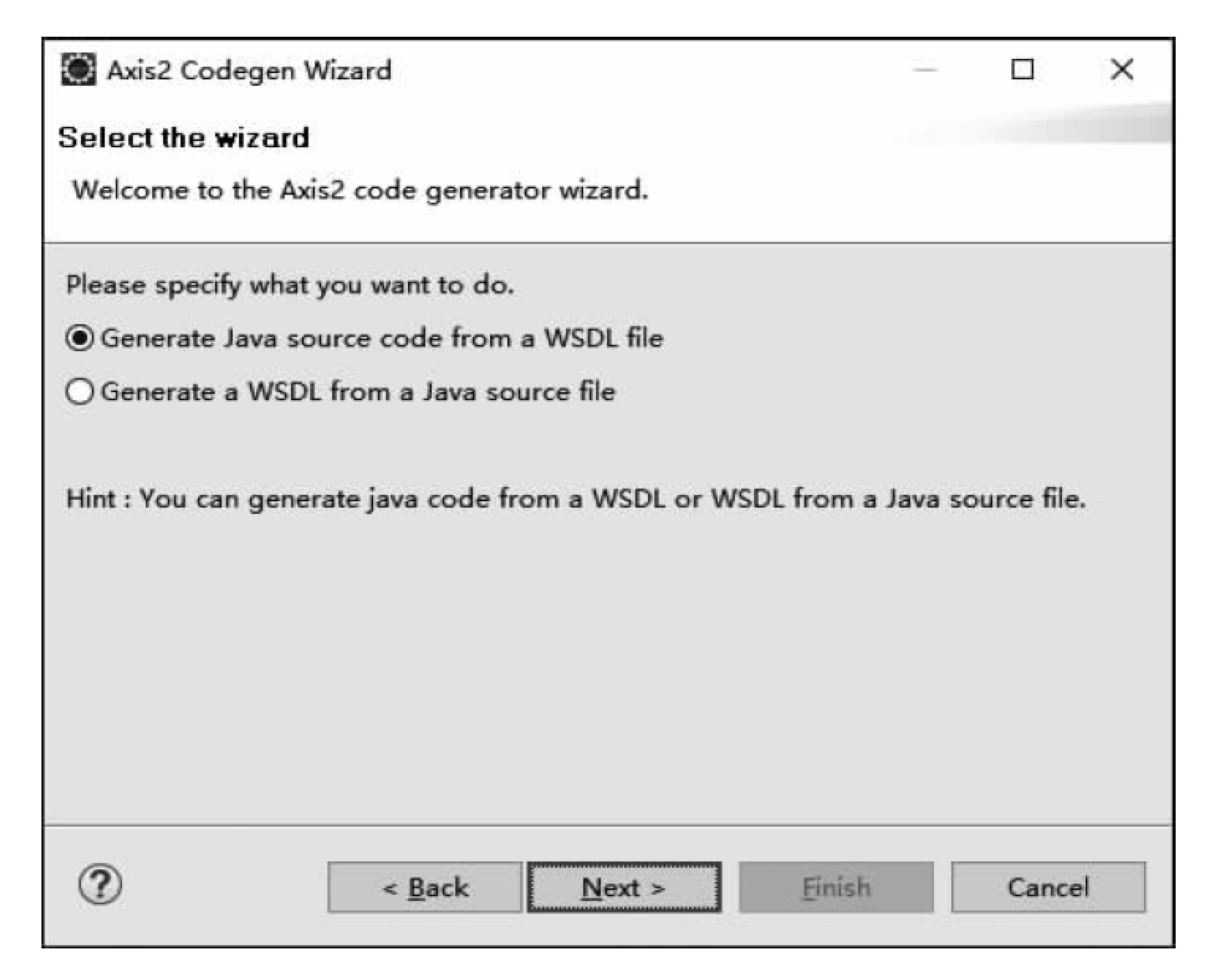


图 7-4　生成 Web Service 客户端代码

此处是调用已有的 Web Service 接口，所以选择根据已有的 WSDL 文件生成 Java 代码，即第一个选项。单击 Next 按钮，输入天气预报 Web Service 的 WSDL 地址"http://ws.webxml.com.cn/WebServices/WeatherWS.asmx?wsdl"，如图 7-5 所示。单击 Next 按钮进入之后的界面，为生成的代码选择一个合适保存路径。



图 7-5　输入 WSDL 地址

（4）新建一个 Java 工程，并将生成完的代码导入新建的工程中，再将第（1）步中下载的"WAR distribution"的 Jar 包也导入工程中。项目结构如图 7-6 所示。

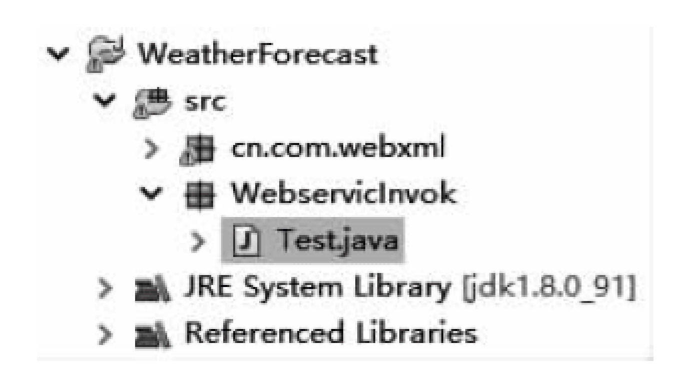


图 7-6　Java 项目结构

（5）编写如下代码。

```
package WebservicInvok;

import java.rmi.RemoteException;
import cn.com.webxml.GetSupportCityString;
import cn.com.webxml.GetSupportCityStringResponse;
import cn.com.webxml.GetWeather;
import cn.com.webxml.GetWeatherResponse;
import cn.com.webxml.WeatherWSStub;

public class Test {
    public static void main(String[] args) throws RemoteException {
        //新建 stub 实例
        WeatherWSStub stub=new WeatherWSStub();

        //取得地区的代码
        GetSupportCityString gr=new GetSupportCityString();
        gr.setTheRegionCode("北京");
        GetSupportCityStringResponse aa=stub.getSupportCityString(gr);
        for(String str : aa.getGetSupportCityStringResult().getString()){
            System.out.println(str);
        }
        System.out.println("===================================");
        //取得天气预报数据
        GetWeather CQWeather=new GetWeather();
        CQWeather.setTheCityCode("792");
        GetWeatherResponse weather=stub.getWeather(CQWeather);
        for(String str : weather.getGetWeatherResult().getString()){
            System.out.println(str);
        }
    }
}
```

（6）程序运行后得到天气预报的结果（见图 7-7）。

2. 在 C# 中调用 Web Service

【例 7-3】 C# 中调用 Web Service。

（1）在 Visual Studio 中创建名为 WeatherForecast 的控制台工程，目录结构如图 7-8 所示。

直辖市 北京
北京
792
2016/11/15 14:28:05
今日天气实况：气温：10℃；风向/风力：南风 2级；湿度：26%
紫外线强度：中等。空气质量：中。
紫外线指数：中等，涂擦SPF大于15、PA+防晒护肤品。
感冒指数：较易发，温差较大，较易感冒，注意防护。
穿衣指数：较冷，建议着厚外套加毛衣等服装。
洗车指数：较适宜，无雨且风力较小，易保持清洁度。
运动指数：较不宜，推荐您进行各种室内运动。
空气污染指数：中，易感人群应适当减少室外活动。



图 7-7　天气预报结果

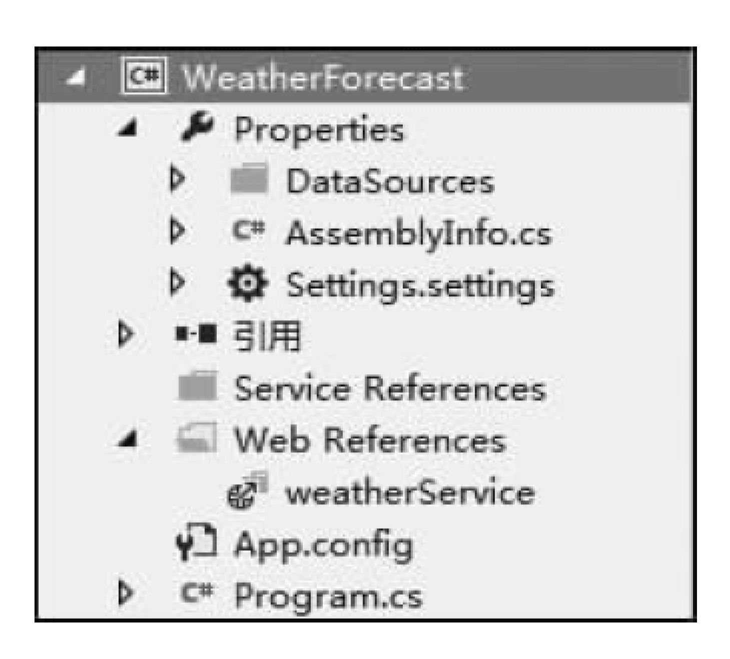


图 7-8　C# 项目结构

（2）右击解决方案，在弹出菜单中选择“添加”→“服务引用”，在弹出的“添加服务引用”界面中单击“高级(V)...”按钮，再弹出的“服务引用设置”界面中单击“添加 Web 引用(W)...”按钮，在弹出的“添加 Web 引用”界面中进行如图 7-9 所示的设置，URL 地址为 http://www.webxml.com.cn/WebServices/WeatherWebService.asmx，设置完成后，单击“确定”按钮。

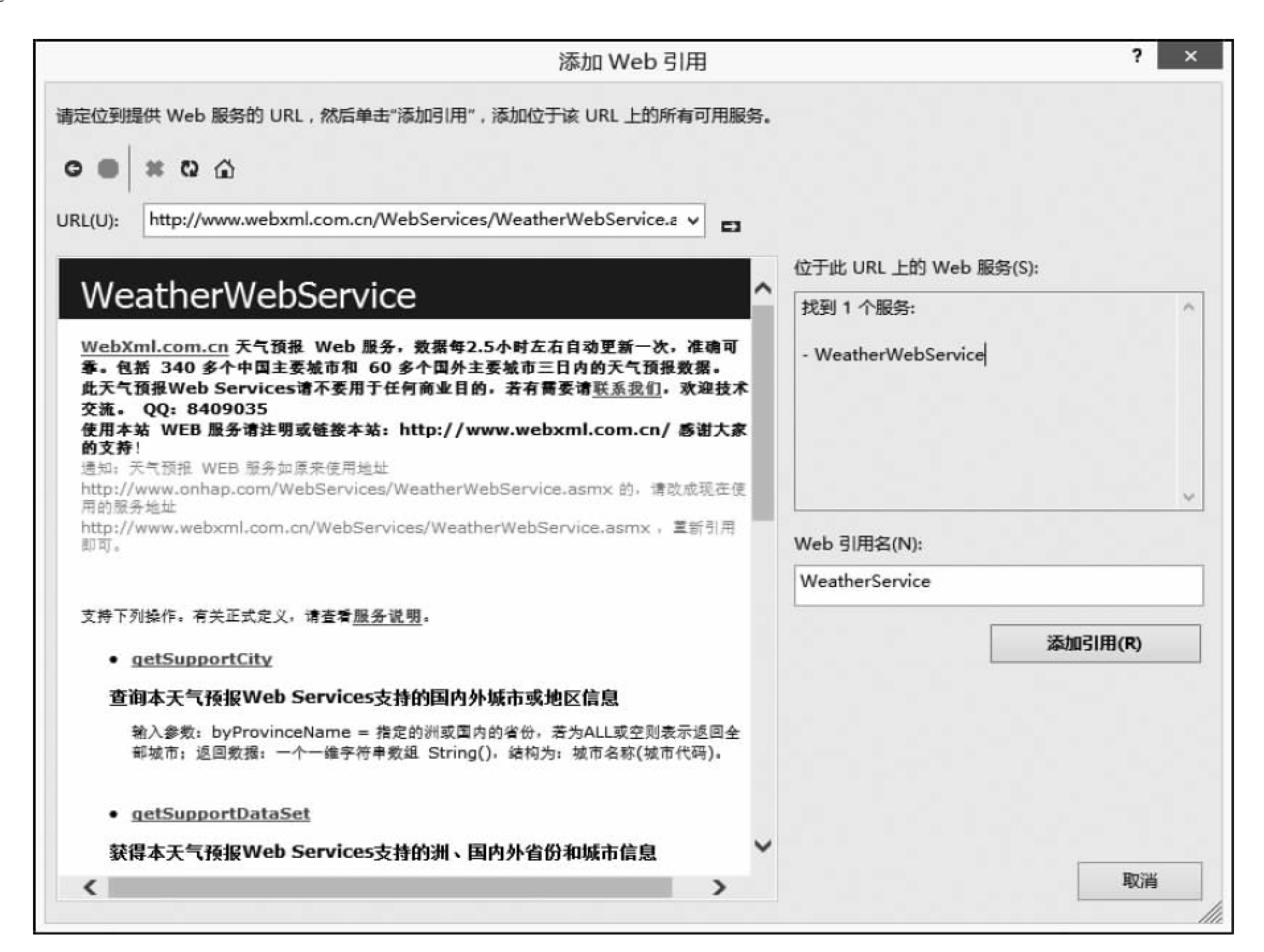


图 7-9　添加 Web 引用

（3）在 Program.cs 文件中编写如下代码。

```
namespace WeatherForecast
{
    class Program
```

```
    {
        static void Main(string[] args)
        {
            //实例化 WeatherWebService
            WeatherService.WeatherWebService ws=new WeatherService.
            WeatherWebService()
            //取得天气预报信息
            IList<string>forecastInfo=ws.getWeatherbyCityName("重庆");
            //输出结果
            foreach (var info in forecastInfo)
            {
                Console.WriteLine(info);
            }
            Console.ReadKey();
        }
    }
}
```

（4）程序运行后得到天气预报的结果，如图 7-10 所示。

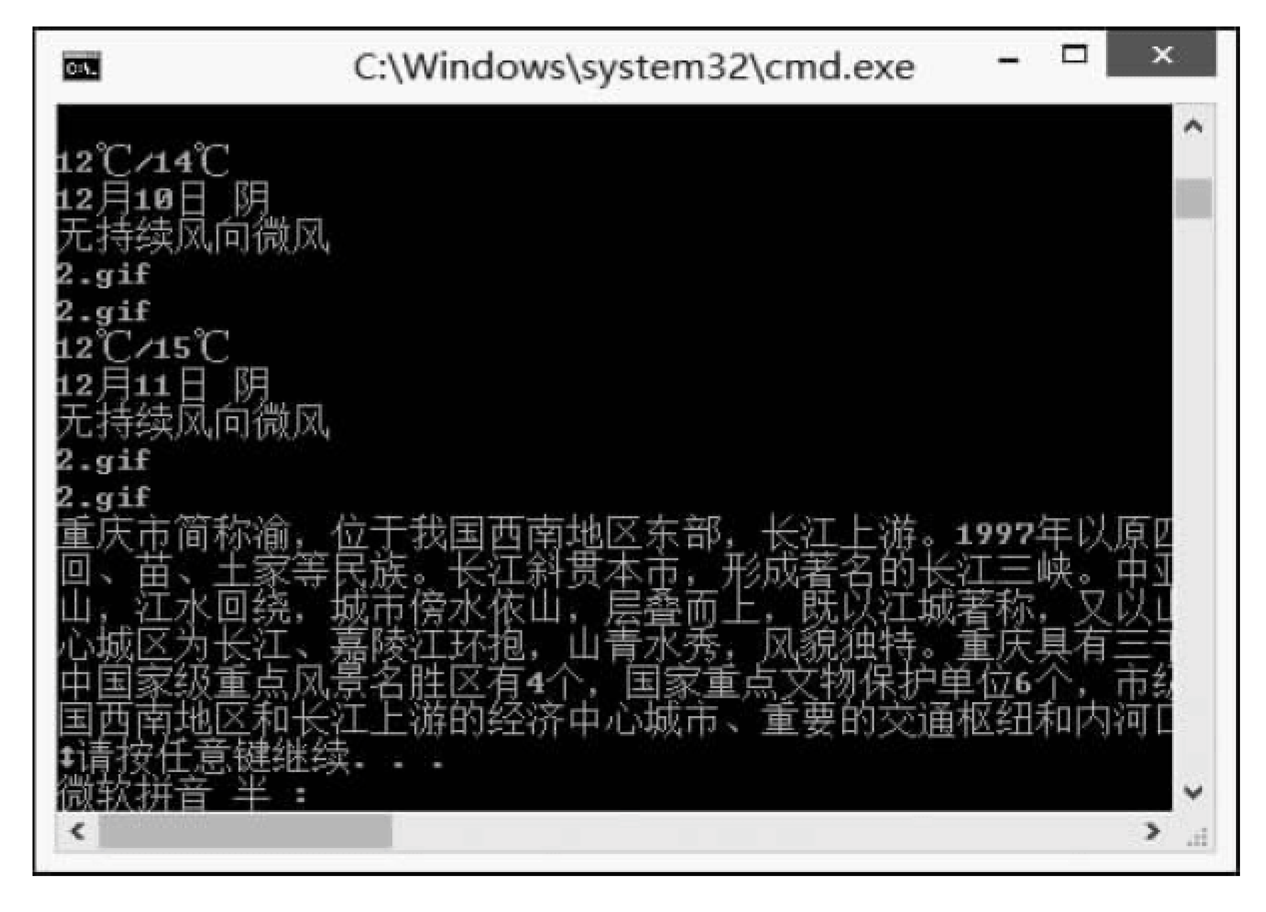


图 7-10　天气预报

7.3　HTTP 请求

HTTP(hypertext transfer protocol，超文本传输协议)是一套计算机通过网络进行通信的规则，该协议用来传输网页、图像以及因特网上在浏览器与服务器之间知道如何处理的其他类型文件。

HTTP 工作于客户端—服务端架构之上。浏览器作为 HTTP 客户端通过 URL 向 HTTP 服务端即 Web 服务器发送所有请求。Web 服务器有 Apache 服务器、IIS 服务器(Internet information services)等。Web 服务器根据接收到的请求向客户端发送响应信息。HTTP 默认端口号为 80，但是也可以改为 8080 或者其他端口。

HTTP 是一种无状态的协议，无状态是指 Web 浏览器和 Web 服务器之间不需要建立持久的连接，这意味着当一个客户端向服务器端发出请求（request），然后 Web 服务器返回响应（response），连接就被关闭了，在服务器端不保留连接的有关信息。HTTP 遵循请求（request）/应答（response）模型。Web 浏览器向 Web 服务器发送请求，Web 服务器处理请求并返回适当的应答。所有 HTTP 连接都被构造成一套请求和应答。

7.3.1 HTTP 通信机制

HTTP 通信机制是在一次完整的 HTTP 通信过程中得以运用，Web 浏览器与 Web 服务器之间将完成下列 4 个步骤。

1. 建立 TCP 连接

在 HTTP 工作开始之前，Web 浏览器首先要通过网络与 Web 服务器建立连接，首先要进行域名的解析，解析完成之后，得到了服务器的 IP 地址，再建立连接。该连接是通过 TCP 来完成的，TCP 协议与 IP 协议共同构建成 Internet，即 TCP/IP 协议族，因此 Internet 又被称作 TCP/IP 网络。HTTP 是比 TCP 更高层次的应用层协议，根据规则，只有低层协议建立之后才能进行更高层协议的连接，如图 7-11 所示。

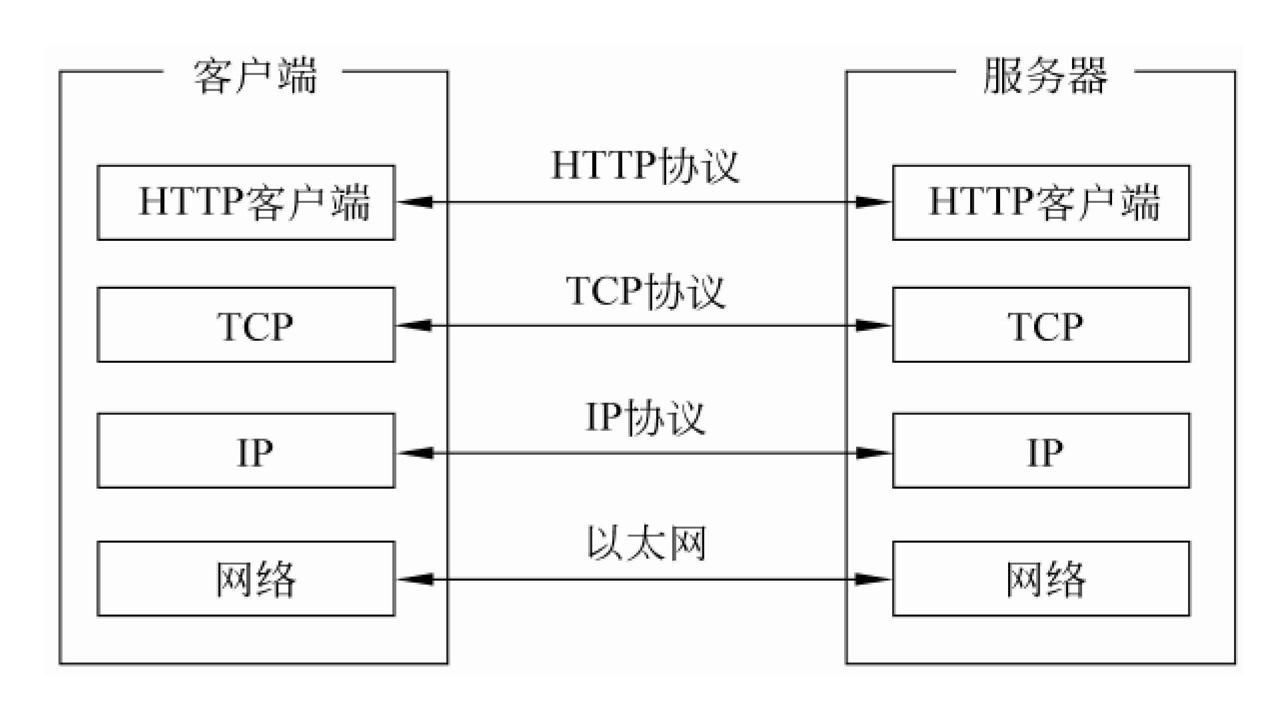


图 7-11　建立 TCP 连接

2. 发起 HTTP 请求

一旦建立了 TCP 连接，Web 浏览器就可以向 Web 服务器发送 HTTP 请求。HTTP 请求信息就是一个数据块，如下所示。

```
POST /login.jsp HTTP/1.1
Accept:image/gif.image/jpeg, * / *
Accept-Language:zh-cn
Accept-Encoding: gzip, deflate
Connection:Keep-Alive
Host:localhost
User-Agent:Mozila/4.0(compatible;MSIE5.01;Window NT5.0)

username=user01 &password=1234
```

从以上定义中可以看出，HTTP 请求信息由请求头和请求数据两部分组成。应特别注意的是，这两部分之间必须要用有一个空行，用于通知服务器该行以下部分是请求数据。

HTTP 请求头包含请求行和请求报头，以上定义的第 1 行就是请求行，该行由请求方法字段、URL 字段和 HTTP 协议版本字段 3 个字段组成，它们用空格分隔。以上定义的第 1 行的意思就是使用 HTTP 1.1 的协议，采用 POST 的方法访问登录页面。其中 GET 和 POST 方法是最常见的请求方法。

使用 GET 方法，请求参数和对应的值附加在 URL 后面，利用一个问号（“?”）代表 URL 的结尾与请求参数的开始，参数之间用“&”符号分隔，一般情况下使用了这种请求方式后，就不需要请求数据部分了。由于是采取在 URL 地址后附加参数的形式，所以传递的参数长度是受限制的，且是明码的传递，安全性不高，常用于一些不涉及系统安全的情况。如果以上定义的 HTTP 请求改成 GET 的请求方式，请求行应该改为如下的定义。

```
GET/login.jsp?username=user01&password=1234 HTTP/1.1
```

使用 POST 方法，可以允许客户端给服务器提供较多信息。POST 方法将请求参数封装在 HTTP 请求数据中（以上定义的第 9 行），以名称/值的形式出现（数据之间也是使用“&”符号分隔），可以传输大量数据，这样 POST 方式对传送的数据大小没有限制，而且也不会显示在 URL 中。为提高传输的数据长度和安全性，通常在表单提交时采用 POST 方法。

请求头部（以上定义的第 2～7 行），由关键字/值对组成，每行一对，关键字和值用英文冒号“:”分隔。请求头部通知服务器有关客户端请求的信息，一些典型的参数如表 7-1 所示。

表 7-1　常见的 HTTP 请求头

状　态　码	说　　明
Accept	客户端可识别的内容类型列表
Accept-Language	指定一种自然语言
Accept-Encoding	指定可接受的内容编码
Connection	当前连接是否保持
Host	请求的主机名，允许多个域名同处一个 IP 地址，即虚拟主机

3. 响应 HTTP 请求

当 Web 服务器接收到 Web 浏览器的请求后，Web 服务器就会根据请求做相应的动作并将结果返回给 Web 浏览器，正如客户端会随同请求发送关于自身的信息一样，服务器也会向 Web 浏览器发送关于它自己的数据及被请求的文档。如下给出了 HTTP 响应的示例。

```
HTTP/1.1 200 OK
Server:Apache Tomcat/5.0.12
Date:Mon,6Oct2003 13:23:42 GMT
Content-Length:112

<html>
<head>
    <title>订餐系统</title>
</head>
<body>
```

```
    <!--欢迎登录订餐管理-->
</body>
</html>
```

由以上定义可以看出，HTTP响应信息也是由两部分构成，即响应头和响应数据。它们之间也是采用与请求信息一样的空行来分隔。对比请求和响应的定义，会发现它们真正的区别在于响应中的第一行用状态信息代替了请求信息。

HTTP响应头包含状态行和响应报头，以上定义的第1行就是状态行，该行由服务器HTTP协议的版本字段、服务器发回的响应状态代码字段、状态代码的文本描述字段3个字段组成，它们之间也用空格分隔。以上定义的第1行的意思就是服务器使用HTTP 1.1协议返回的响应代码为200，并成功(OK)处理了用户请求的响应信息。响应状态代码由三位数字组成，第一个数字定义了响应的类别，有五种类别，如表7-2所示。

表7-2　HTTP响应码的类别

状态码	说　明
1××	指示信息：表示请求已接收，并会继续处理
2××	成功：表示请求已被成功接收、理解、接受
3××	重定向：要完成请求，必须进行更进一步的操作
4××	客户端错误：请求有语法错误或请求无法实现
5××	服务器端错误：服务器未能实现合法的请求

每个类型的状态码都对应很多的具体状态，比较常用的如表7-3所示。

表7-3　常见的HTTP响应码

状态码	描　述	说　明
200	OK	客户端请求成功
400	Bad Request	客户端请求有语法错误，不能被服务器所理解
404	Not Found	请求资源不存在。举个例子：输入了错误的URL
500	Internal Server Error	服务器发生不可预期的错误

响应报头(响应定义的第2～4行)，也是由关键字/值对组成，每行一对，关键字和值也用英文冒号“:”分隔。响应头部也和请求头一样包含许多服务器响应相关的有用信息，例如服务器类型、日期时间、内容类型和长度等。该部分是可以在响应信息中省略的。

响应数据(响应定义的第6～13行)，该部分信息由Web服务器将处理结果封装为HTML后再返回给Web浏览器。Web浏览器在接收到响应数据之后，对整个页面进行刷新显示。

4. 关闭TCP连接

一般情况下，Web服务器向浏览器发送了响应信息之后就要关闭TCP连接。在HTTP 1.1中，如果在Web浏览器的请求头部信息中指定Connection属性为keep-alive，TCP连接在发送后将仍然保持打开的状态，浏览器就可以继续通过相同的连接发送请求，这样节省了为每个请求建立新连接所需的时间，还可节约网络带宽。

7.3.2　HTTP 请求的调用

在 7.2.3 小节中使用后台代码取得了 Web Service 中的天气预报数据，但是并没有显示到 Web 页面中，本小节将分别用 Java 的动态网页 JSP 和 ASP.NET 的中动态网页 ASPX 取得天气预报的数据并展示到前台 Web 页面中。

1. JSP 页面的 HTTP 请求

【例 7-4】 JSP 页面中的 HTTP 请求。

(1) 在 Eclipse 中新建名为 WeatherForecastWeb 的动态 Web 工程，并将相关的 Web Service 的包导入 WEB-INF 的 lib 目录中，项目的目录结构如图 7-12 所示。

(2) 在 Servlet 包中新建名为 GetWeatherForecast.java 的 Servlet，并在 doGet 方法中输入以下代码。



图 7-12　JSP 下的 HTTP 请求的目录

```
protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOException {
    request.setCharacterEncoding("utf-8");
    response.setContentType("text/html;charset=utf-8");

    //获取表单数据
    String cityid=request.getParameter("cityid");

    //新建 stub 实例
    WeatherWSStub stub=new WeatherWSStub();

    //取得天气预报数据
    GetWeather Weather=new GetWeather();
    Weather.setTheCityCode(cityid);
    GetWeatherResponse weather=stub.getWeather(Weather);
    PrintWriter out=response.getWriter();

    //循环输出结果
    String WeatherResult="<table>";
    for(String str : weather.getGetWeatherResult().getString()){
        WeatherResult+="<tr><td>"+str+"</td></tr>";
    }
    WeatherResult+="</table>";
    out.print(WeatherResult);
}
```

(3) 在 WebContent 文件夹下新建名为 index.jsp 的 JSP 文件。

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding
="UTF-8"%>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>取得天气预报</title>
</head>
<body>
<form action="GetWeatherForecast"  method="post">
        <select id="cityid" name="cityid" style="width:180px;">
            <option selected>=请选择城市类型=</option>
            <option value="792">北京</option>
            <option value="1599">重庆</option>
        </select>
        <input type="submit" value="查询"/>
</form>
</body>
</html>
```

(4) 运行 index.jsp 页面,并选择“北京”。单击“查询”按钮,运行效果如图 7-13 所示。

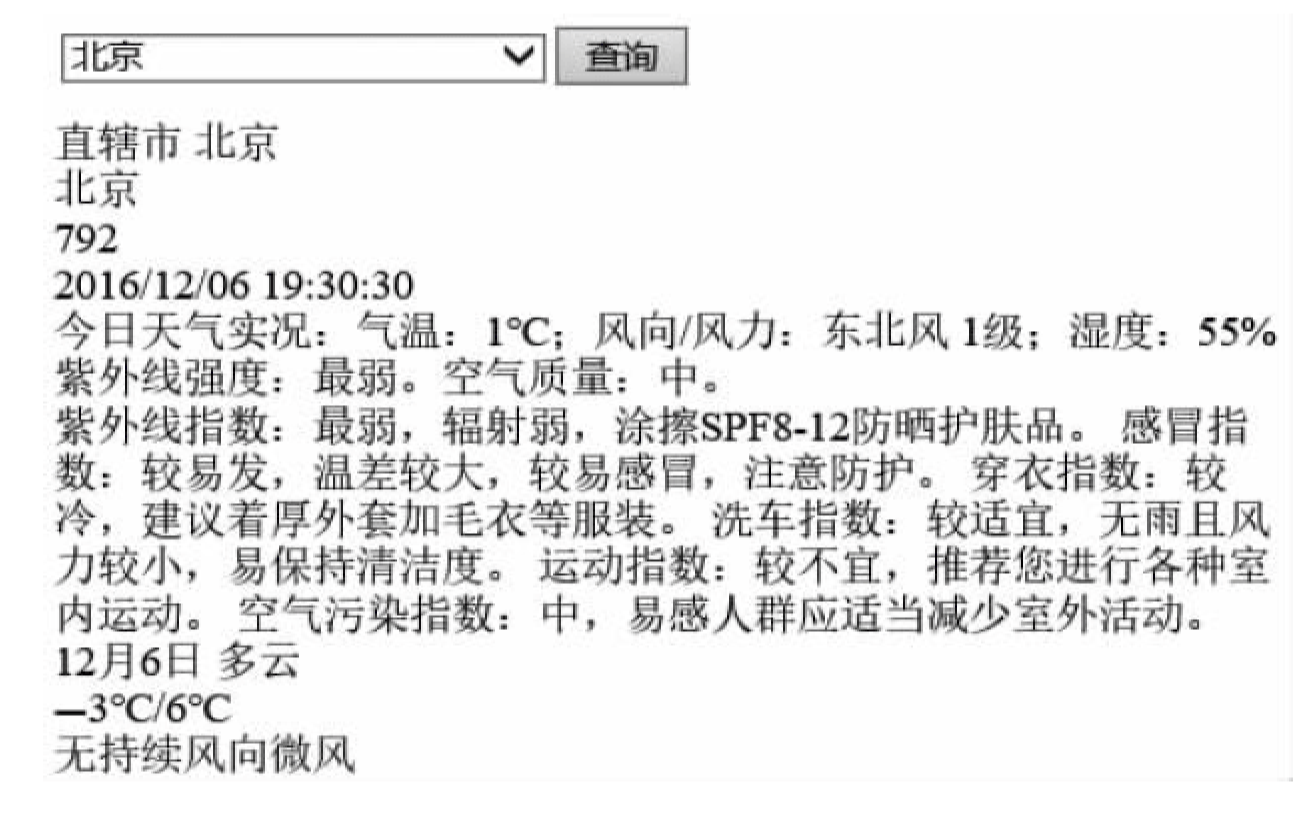


图 7-13　JSP 的 HTTP 请求

(5) 在 IE 的“开发人员工具”中查看 HTTP 请求。在 IE 浏览器中选择“工具”→“开发人员工具”菜单,该工具提供了各种网页调试的便利工具。在弹出的“工具框”中选择“网络”选项卡后,在页面上进行查询操作,就能看到如图 7-14 所示的 HTTP 请求信息。

拖动右边的滚动条能看到请求标头的信息。选择“正文”选项卡,能看到 HTTP 的响应正文和请求正文,如图 7-15 所示。

2. ASP.NET 页面中的 HTTP 请求

【例 7-5】 ASP.NET 页面中的 HTTP 请求。

(1) 在 Visual Studio 中创建名为 WeatherForecastWeb 的 Web 工程,并引入天气预报的 Web Service,具体目录结构如图 7-16 所示。



图 7-14　在“开发人员工具”中查看 HTTP 请求

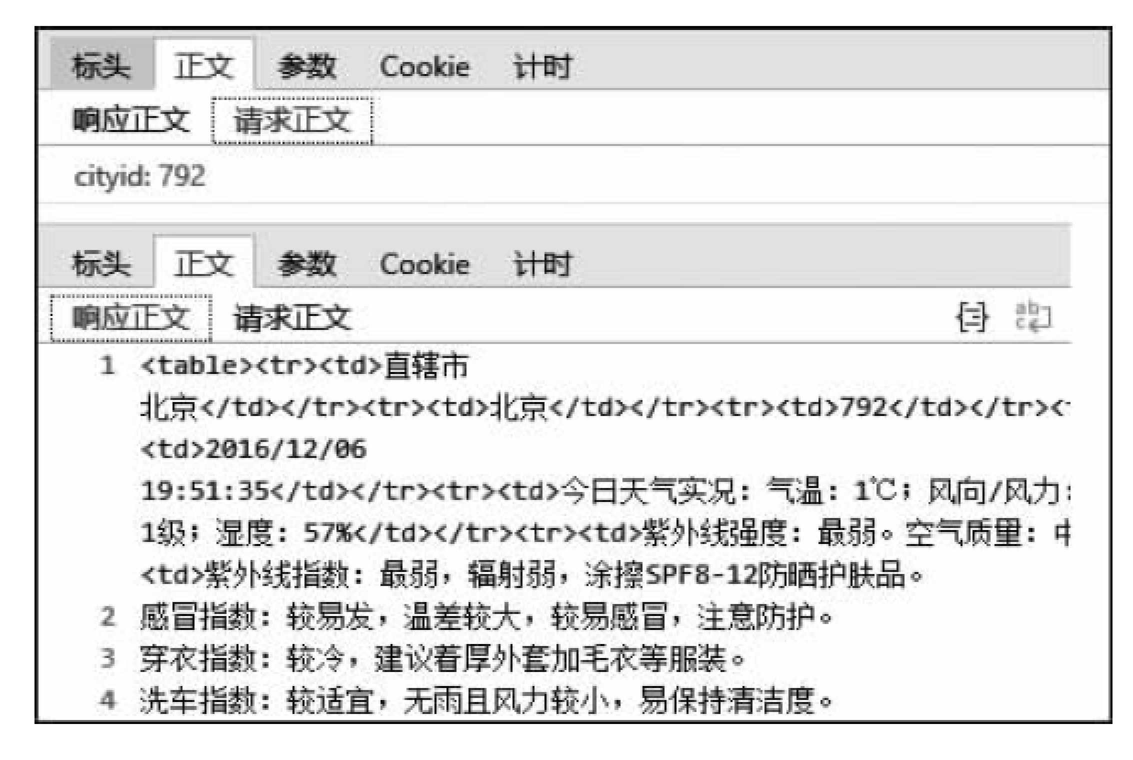


图 7-15　查看 HTTP 请求的正文

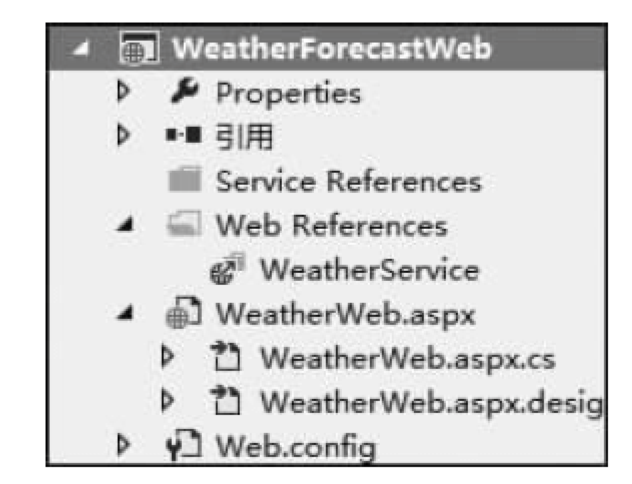


图 7-16　ASP.NET 的 Web 工程目录

（2）在 WeatherWeb.aspx 中编写前台显示代码。

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WeatherWeb.aspx.cs"
        Inherits="WeatherForecastWeb.WeatherWeb" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:DropDownList ID="ddCity" runat="server">
                <asp:ListItem>重庆</asp:ListItem>
                <asp:ListItem>北京</asp:ListItem>
            </asp:DropDownList>
            <asp:ButtonID="btnClick" runat="server"
                Text="天气查询" OnClick="btnClick_Click" />
```

```
            <p id="txtShow" runat="server"></p>
        </div>
    </form>
</body>
</html>
```

(3) 在 WeatherWeb.aspx.cs 中编写后台显示代码。

```
namespace WeatherForecastWeb
{
    public partial class WeatherWeb: System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
        }
        protected void btnClick_Click(object sender, EventArgs e)
        {
            WeatherService.WeatherWebService ws=new WeatherService.
            WeatherWebService();
            string cityName=this.ddCity.SelectedValue;
            IList<string>list=ws.getWeatherbyCityName(cityName);
            foreach (var item in list)
            {
                this.txtShow.InnerHtml+=item+"<br />";
            }
        }
    }
}
```

(4) 运行该页面并选择“北京”,单击“查询”按钮,运行效果如图 7-17 所示。

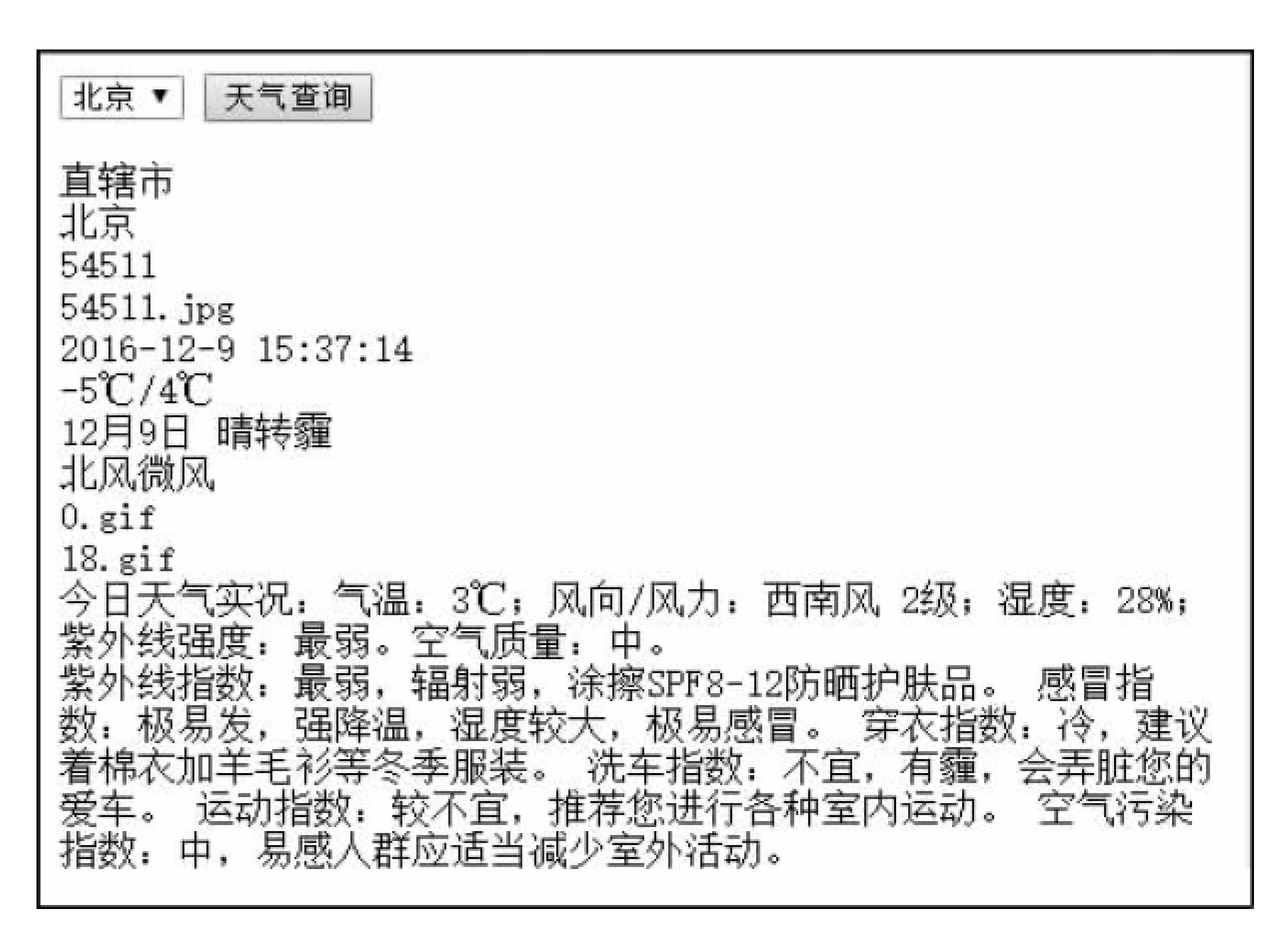


图 7-17　ASP.NET 的 HTTP 请求

以上请求方法中,关于在浏览器中查看 HTTP 请求的方法请参考例 7-4 的第(5)步。

7.4　ajax

ajax(asynchronous JavaScript and XML,异步 JavaScript 和 XML)是一种创建交互式网页应用的网页开发技术。相对于采用 HTTP 请求的传统网页更新内容时重载整个网页页面而言,采用 ajax 技术的网页不必中断交互,完整地刷新页面,就可以动态地更新 Web 页面。这里的异步是相对于 HTTP 请求流程而言的,HTTP 请求是顺序执行的,是同步的。将对 Web 服务器的请求和数据加载从当前浏览器页面独立出来并单独执行就是异步请求,它通过 JavaScript 接收服务器传来的数据,然后操作 DOM 对网页的某部分进行更新。使用 ajax 技术最直观的感受是向服务器获取新数据时不需要刷新页面等待了,使用它可以创建更加丰富、更加动态的 Web 应用程序用户界面,其即时性与用户体验感甚至能够接近桌面应用程序。

ajax 不是一种新的编程语言,它是多种技术的综合,包括 JavaScript、XHTML 和 CSS、DOM、XML 和 XSTL、XMLHttpRequest。其中,使用 XHTML 和 CSS 可进行标准化呈现,使用 DOM 可以实现动态显示和交互,使用 XML 和 XSTL 可以进行数据交换与处理,使用 XMLHttpRequest 对象可以进行异步数据读取,使用 JavaScript 可以绑定和处理所有数据。

7.4.1　XMLHttpRequest 对象

XMLHttpRequest(可扩展超文本传输请求,简称 XHR)对象是 ajax 的核心对象。它提供了对 HTTP 协议的完全访问,包括做出 POST 和 HEAD 请求以及普通的 GET 请求的能力。XMLHttpRequest 对象可以异步地返回 Web 服务器的响应,并且能以文本或者一个 DOM 文档形式返回内容。尽管名为 XMLHttpRequest,它并不限于和 XML 文档一起使用,它可以接收任何形式的文本文档。它为向服务器发送请求和解析服务器响应提供了流畅的接口。通过这个对象,ajax 可以像桌面应用程序一样与 Web 服务器进行数据的交换,而不用每次都刷新界面,也不用每次将数据处理的工作都交给服务器来做;这样既减轻了服务器负担又加快了响应速度,缩短了用户等待的时间。

1. 创建 XMLHttpRequest 对象

IE 5.0 开始,开发人员在 Web 页面中使用的 XMLHttpRequest 对象是通过 MSXML 库中一个 ActiveX 对象实现的,IE 7 以上版本及其他较新的浏览器(Firefox、Chrome、Safari 以及 Opera 等)均内建了 XMLHttpRequest 对象,XMLHttpRequest 对象得到了现代所有浏览器较好的支持。在这些浏览器中只需使用 XMLHttpRequest 构造函数就可以构造 XMLHttpRequest 对象。如下的代码用于构建兼容不同浏览器的 XMLHttpRequest 对象:

```
<script language="javascript" type="application/javascript">
    var xmlhttp;
    if(window.XMLHttpRequest)  {
```

```
        //适合 IE 7+、Firefox、Chrome、Opera、Safari 的代码
        xmlhttp=new XMLHttpRequest();
    }else{
        //适合 IE 6 及 IE 5 的代码
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
</script>
```

2. 向服务器发送请求

在创建好 XMLHttpRequest 对象之后，我们可以使用 XMLHttpRequest 对象的 open()和 send()方法向服务器发送请求。open()用于初始化 HTTP 请求参数，但并不发送请求；send()用于向服务器发送 HTTP 请求。它们的语法规则如表 7-4 所示。

表 7-4 XMLHttpRequest 对象的 open()和 send()方法

方　法	描　述
open(method,url,async)	规定请求的类型、URL 以及是否异步处理请求。 method：请求的类型为 GET 或 POST。 url：文件在服务器上的位置。该文件可以是任何类型的文件，比如.txt 和.xml；或者服务器脚本文件，比如.asp 和.php(在传回响应之前，能够在服务器上执行任务)。 async：true(异步)或 false(同步)。如果要用于 ajax，必须设置为 true
send(string)	将请求发送到服务器。参数 string 仅用于 POST 请求

类似于 HTTP 请求，XMLHttpRequest 对象也能使用 GET 和 POST 的方法。

(1) 使用 GET 的形式向服务器发送请求

以不带参数的 GET 方式发送请求，如下所示。

```
<script language="javascript" type="application/javascript">
    xmlhttp.open("GET","demo_get.asp",true);
    xmlhttp.send();
</script>
```

采取这样的方式取得的结果可能是本地的缓存结果，为了避免这种情况，可以在 URL 地址后附加一个随机数，强制让服务器每次都到服务器上去取得最新的数据。代码如下所示。

```
<script language="javascript" type="application/javascript">
    xmlhttp.open("GET","demo_get.asp? t="+Math.random(),true);
    xmlhttp.send();
</script>
```

如果要向服务器发送其他带参数的请求，代码和以上的示例类似，只是变量名要和服务器端的变量一致。

(2) 使用 POST 的形式向服务器发送请求

当需要向服务器发送大量数据或者发送的数据中包括未知字符的用户输入时，需要用

到 POST 方式发送请求，代码如下所示。

```
<script language="javascript" type="application/javascript">
    xmlhttp.open("POST","ajax_test.asp",true);
    xmlhttp.setRequestHeader("Accept-Language "," zh-cn");
    xmlhttp.send("fname=Bill&lname=Gates");
</script>
```

其中，setRequestHeader(header，value)方法的值取的是 HTTP 请求的头部的信息，将名称/值的形式对应到 header、value 字段中。

(3) onreadystatechange 事件

当使用的 XMLHttpRequest 对象的 open()方法的 async 参数设置为 true 时，就需要使用 XMLHttpRequest 的 onreadystatechange 事件来监听请求的执行情况，它的典型代码如下所示。

```
<script language="javascript" type="application/javascript">
    xmlhttp.onreadystatechange=function(){
        if(xmlhttp.readyState==4 && xmlhttp.status==200){
            //TODO 对服务器应答内容进行操作并显示
        }
    }
</script>
```

以上代码的第 3 行的 readyState 属性保存有 XMLHttpRequest 对象的状态信息，当一个 XMLHttpRequest 对象被创建时，这个属性值从 0 开始，直到接收到完整的 HTTP 响应，这个值增加到 4。它们的含义如表 7-5 所示。

表 7-5　readyState 属性

状态代码	描　述
0	XMLHttpRequest 对象已被创建或已被 abort()方法重置
1	open()方法已调用，但 send()方法还没被调用，请求尚未发送
2	send()方法已调用，HTTP 请求已发送到服务器，但还没有收到响应
3	所有响应头部已经收到。响应体开始接收但尚未完成
4	HTTP 响应已经完全接收

readyState 属性值不会递减，除非当一个请求在处理过程中调用了 open()或 abort()方法。每次这个属性值增加时，都会触发 onreadystatechange 事件句柄。一次完整的请求过程状态由 0 到 4，onreadystatechange 事件就被触发了 5 次。

第 3 行的 status 属性是服务器返回的 HTTP 响应码(参考表 7-3)，当为 200 时，表示读取返回值成功。

所以，当 readyState 等于 4 且 status 等于 200 时，表示响应已经就绪，就可以在代码中处理返回的内容了。

3. 获取响应内容

XMLHttpRequest 对象有两个属性来存储响应的内容：responseText 和 responseXML，

如表 7-6 所示。

表 7-6　responseText 和 responseXML

属　　性	描　　述
responseText	获得字符串形式的响应数据
responseXML	获得 XML 形式的响应数据

4. XMLHttpRequest 对象的使用

现在以 XML 文件作为 Web 服务器的响应内容,来完成一个完整的 XMLHttpRequest 对象的请求实例,如例 7-6 所示。

【例 7-6】 XMLHTTPRequest 对象请求 XML 文件。

(1) 建立如下的餐厅菜单的 XML 文件。

```
<?xml version="1.0" encoding="UTF-8"?>
<menu>
    <cuisine id="001">
        <name>宫保鸡丁</name>
        <price>20.00</price>
    </cuisine>
    <cuisine id="002">
        <name>京酱肉丝</name>
        <price>28.00</price>
    </cuisine>
    <cuisine id="003">
        <name>乱炖</name>
        <price>32.00</price>
    </cuisine>
    <cuisine id="004">
        <name>重庆小面</name>
        <price>7.00</price>
    </cuisine>
</menu>
```

(2) 创建 HTML 代码,如下所示。

```
<html>
<head>
<script type="text/javascript">
    function loadXMLDoc() {
        var xmlhttp;
        var txt, x, i;
        if(window.XMLHttpRequest) {
            //代码适用于 IE 7+、Firefox、Chrome、Opera、Safari
            xmlhttp=new XMLHttpRequest();
        } else {
            //代码适用于 IE 6、IE 5
            xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
        }
```

```
        xmlhttp.onreadystatechange=function() {
            if(xmlhttp.readyState==4 && xmlhttp.status==200) {
                xmlDoc=xmlhttp.responseXML;
                txt="";
                x=xmlDoc.getElementsByTagName("name");
                for(i=0; i<x.length; i++) {
                    txt=txt+x[i].childNodes[0].nodeValue+"<br />";
                }
                document.getElementById("myDiv").innerHTML=txt;
            }
        }
        xmlhttp.open("GET", "menu.xml?t="+Math.random(), true);
        xmlhttp.send();
    }
</script>
</head>

<body>
    <h2>利用 XMLHTTPRequest 对象取得 XML 数据:</h2>
    <button type="button" onclick="loadXMLDoc()">取得餐厅的菜单列表</button>
    <div id="myDiv"></div>
</body>
</html>
```

（3）程序的运行效果如图 7-18 所示。

利用XMLHttpRequest对象取得XML数据:

取得餐厅的菜单列表
宫保鸡丁
京酱肉丝
乱炖
重庆小面

图 7-18　利用 XMLHttpRequest 对象取得 XML 数据

注意：本例的代码必须放在 IIS 或者 Tomcat 中运行，否则不能取得 XMLHttpRequest 对象 status 属性的值。

7.4.2　JSON

在 Web 浏览器和 Web 服务器进行交互的过程中，可能需要将服务器的对象传递到浏览器中，也可能需要把浏览器的对象传递到服务器上。在 Web 浏览器中主要用 JavaScript，在后台逻辑程序中使用 Java、C＃、PHP 等后台语言。如何实现这两种不同类型语言之间的信息传递呢？使用之前讲到的 XML 文件是一个不错的选择，但是这种方式需要浏览器和服务器在每次通信过程中都要对 XML 文件进行解析。如果有一种形式能直接将对象进行传递，将为程序员减少不少的工作量。

JSON(JavaScript object notation，JavaScript 对象表示法)是一种轻量级的具有自我描述性的数据交换格式。它基于 ECMAScript 的一个子集。JSON 采用完全独立于语言的文本格式。这些特性使 JSON 成为理想的数据交换语言。它可以将 JavaScript 对象中表示的

一组数据转换为字符串，然后就可以在 JavaScript 函数之间轻松地传递这个字符串，或者在异步应用程序中将字符串在 Web 客户机和 Web 服务器端程序之间来回传递。

1. JSON 的语法

JSON 的书写规则为“名称/值对”的形式，中间用“:”相连接。类似于 HTTP 请求的请求头部的表示方式，但是在这里名称和值都必须用双引号包括。JSON 语法如下：

```
{ "name":"宫保鸡丁"}
```

以上定义可以理解为 name="宫保鸡丁"。JSON 值可以是数字(整数或浮点数)、字符串(string)、逻辑值(true 或 false)、数组(在方括号中)、对象(在花括号中)或者 null。一个“名称/值对”没有确切的含义。当多个“名称/值对”串在一起时就能体现 JSON 的价值。“名称/值对”之间用“,”连接，且将它们放置在一个“{}”之中就表示一个对象。JSON 对象的表示如下所示。

```
{"id":001, "name":"宫保鸡丁","price":20.00}
```

将一组 JSON 对象表示在数组中时，就成为 JSON 对象数组。JSON 对象数组的表示和 JavaScript 的数组表示一样，将对象放置在“[]”之中，对象之间用“,”分隔。JSON 对象数组的表示如下所示。

```
{
    "menu":[
        {"id":"001","name":"宫保鸡丁","price":20.00},
        {"id":"002","name":"京酱肉丝","price":28.00},
        {"id":"003","name":"乱炖","price":32.00},
        {"id":"004","name":"重庆小面","price":7.00}
    ]
}
```

以上 JSON 数组将 7.4.1 小节实例中的 XML 文件转换为了 JSON 的表示方式。对比两种表示方式，JSON 的表示方式相对于 XML 来说更为简洁，计算机通过索引就能读到响应的字段，XML 还要经过文档的解析。从编码阅读性上来说，XML 有规范的标签形式，更利于阅读和理解，JSON 对象要是去掉空白制表符以及换行，JSON 就是密密麻麻的数据，不便于人们的阅读和理解。但是它们都具有很好的扩展性，可以这么说，没有什么是 XML 可以扩展而 JSON 却不能扩展的。不过 JSON 在 JavaScript 中可以存储 JavaScript 复合对象，有着 XML 不可比拟的优势。

2. JSON 在 JavaScript 中的应用

由于 JSON 语法是 JavaScript 语法的子集，因此可以使用 JavaScript 中的全局函数 eval() 来解析 JSON 文本并生成 JavaScript 对象。eval() 函数只有一个参数。如果传入的参数不是字符串，它直接返回这个函数。如果参数是字符串，它会把字符串当成 JavaScript 代码进行编译并执行，所以将 JSON 文本传给 eval() 函数后能得到 JavaScript 的对象。它的使用如例 7-7 所示。

【例 7-7】　在 JavaScript 中使用 JSON。

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta charset="UTF-8">
<script type="text/javascript">
    var txt='{"menu":[{'+
        '"id":"001","name":"宫保鸡丁","price":20.00},'+
        '{"id":"002","name":"京酱肉丝","price":28.00},'+
        '{"id":"003","name":"乱炖","price":32.00},'+
        '{"id":"004","name":"重庆小面","price":7.00}'+
        ']}';
    function loadXMLDoc() {
        var obj=eval("("+txt+")");
        var content="";
        for(i=0; i<obj.menu.length; i++) {
            content=content+obj.menu[i].name+"<br />";
        }
        document.getElementById("myDiv").innerHTML=content;
    }
</script>
</head>
<body>
    <h2>利用 JavaScript 来取得 JSON 对象数据:</h2>
    <button type="button" onclick="loadXMLDoc()">取得餐厅的菜单列表
    </button>
    <div id="myDiv"></div>
</body>
</html>
```

程序的运行效果如图 7-19 所示。



图 7-19　在 JavaScript 中使用 JSON

JSON 对象的优势在异步 Web 应用程序中。关于 JSON 在 ajax 中的使用，在第 8 章的综合项目中将大量采用。详细的例子参考第 8 章的综合项目。

7.4.3　jQuery 中的 ajax

因为不同的浏览器对 ajax 的实现并不相同，编写常规的 ajax 代码并使之在各个浏览器之间兼容并不容易。在实现相应功能的同时，还必须编写额外的代码对不同的浏览器进行测试。不过，jQuery 解决了这个难题，在 jQuery 库中提供了完整的 ajax 兼容套件，只需要

一行简单的代码，就可以实现在各个浏览器之间兼容的 ajax 功能。也就是说，通过 jQuery 的 ajax 方法，能够使用 HTTP 请求的 Get 方式或者 Post 方式从远程服务器上请求文本、HTML、XML 或 JSON 数据。在 jQuery 中 $.ajax()方法属于最底层的函数，第 2 层是 $.load()、$.get()和 $.post()函数，第 3 层是 $.getScript()和 $.getJSON()函数。

1. $.ajax()函数

$.ajax()函数是 jQuery 对 ajax 封装最基础的函数，通过这个函数可以完成异步通信的所有功能。该方法的参数类似于 JSON 文本定义对象的形式，采用“名称/值对”的形式，用“:”相连接，不同的“名称/值对”之间用“,”连接，$.ajax()函数的语法如下所示。

```
$.ajax({name:value, name:value, … })
```

$.ajax()函数中常用的“名称/值对”如表 7-7 所示。

表 7-7　$.ajax()函数中常用的“名称/值对”

名　称	值
type	规定请求的类型(GET 或 POST)
url	规定请求的 URL
async	布尔值，表示请求是否异步处理。默认是 true
data	规定要发送到服务器的数据
dataType	预期的服务器响应的数据类型
error(xhr,status,error)	当请求失败时要运行的函数
success(result,status,xhr)	当请求成功时运行的函数
complete(xhr,status)	请求完成时运行的函数(在请求成功或失败之后均调用，即在 success 和 error 函数之后)

使用 jQuery 的 ajax 不但能够向 Web 服务器端的资源发起请求，还能向 WebService 发起请求。关于 jQuery 中 ajax 的开发实例，在第 8 章的综合项目中有大量的实例，本章以用 $.ajax()函数来取得返回 JSON 对象的天气预报 WebService 的例子来说明 $.ajax()的使用方法，代码如例 7-8 所示。

【例 7-8】　使用 $.ajax()函数取得天气预报。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>jQuery 的 ajax 连接 WebService:</title>
<script type="text/javascript" src="js/jquery.min.js"></script>
<style>
    span {
        display: inline-block;
        width: 150px;
    }
</style>
<script language="javascript" type="application/javascript">
```

```
    $(document).ready(function () {
        $('#btnGetWeather').click(function() {
            var temp="";
            var city=$("#city").val();
            $.ajax({
                type:"POST",
                url:"http://wthrcdn.etouch.cn/weather_mini",
                data:{city:city},
                async:false,
                dataType: "jsonp",
                error:function(){
                    alert("请求出错了。");
                },
                success:function(data){
                        var jsonValue=data;
                        jsonValue.data.forecast.forEach(function(e) {
                        temp=temp+"<span>"+jsonValue.data.city+
                            "</span><span>"+e.date+
                            "</span><span>"+e.fengli+
                            "</span><span>"+e.fengxiang+
                            "</span><span>"+e.high+
                            "</span><span>"+e.low+
                            "</span><span>"+e.type+
                            "</span><br />";
                        $("#myDiv").html(temp);
                        });
                }
            });
        })
    });
</script>
</head>
<body>
    <center>
        <h2>$ajax 调用 WebService 查询天气预报</h2>
        <select id="city">
            <option value="重庆">重庆</option>
            <option value="北京">北京</option>
            <option value="广州">广州</option>
        </select>
        <button type="button" id="btnGetWeather">查询</button>
        <hr />
        <div id="myDiv"></div>
    </center>
</body>
</html>
```

程序的运行效果如图 7-20 所示。

$ajax调用WebService查询天气预报

重庆 查询

重庆	2日星期五	微风级	无持续风向	高温 13℃	低温 10℃	阴
重庆	3日星期六	微风级	无持续风向	高温 13℃	低温 8℃	小雨
重庆	4日星期天	微风级	无持续风向	高温 15℃	低温 8℃	晴
重庆	5日星期一	微风级	无持续风向	高温 16℃	低温 11℃	多云
重庆	6日星期二	微风级	无持续风向	高温 16℃	低温 10℃	阴

图 7-20　使用 jQuery 的 ajax

2. $.load()函数

$.load()函数是 jQuery 中最简单和比较常用的 ajax 方法，它可以请求从服务器加载数据，并把返回的数据放置到指定的元素中。$.load()的语法如下：

```
load(url,parameters,callback)
```

其中，url 是希望请求的 url 地址(HTML 文件的地址)，是字符串类型，且是必须要设定参数；parameters 是规定要发送到服务器的数据，采用“名称/值对”的形式，该项为可选参数；callback 是请求完成时的回调函数，函数 function(data，status，xhr)为 JavaScript 的形式，data 包含了请求的结果数据，status 包含请求的状态，xhr 包含了 XMLHttpRequest 对象，该项也为可选项。但是如果需要对结果进行处理时，该函数不能省略。$.load()函数的应用如例 7-9 所示。

【例 7-9】 使用$.load()函数取得天气预报。

首先创建一个如下的天气预报的 HTML 页面。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
</head>
<body>
    <div id="beijingWeather">
        <div id="FortyEightHours">
        <span>北京</span><span>23 日星期五</span><span>微风级</span><span>南
        风</span>
        <span>高温 3℃</span><span>低温 -6℃</span><span>晴</span><br>
        <span>北京</span><span>24 日星期六</span><span>微风级</span><span>南
        风</span>
        <span>高温 3℃</span><span>低温 -5℃</span><span>多云</span><br>
        </div>
        <span>北京</span><span>25 日星期天</span><span>微风级</span><span>南
        风</span>
        <span>高温 3℃</span><span>低温 -2℃</span><span>霾</span><br>
        <span>北京</span><span>26 日星期一</span><span>微风级</span><span>北
        风</span>
        <span>高温 1℃</span><span>低温 -7℃</span><span>阴</span><br>
```

```
        <span>北京</span><span>27 日星期二</span><span>微风级</span><span>南
        风</span>
        <span>高温 1℃</span><span>低温 -6℃</span><span>晴</span><br>
    </div>
</body>
</html>
```

再创建显示页面，具体的 HTML 代码如下：

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>使用 jQuery 的$load()函数</title>
<script type="text/javascript" src="js/jquery.min.js"></script>
<style>
    span {
        display: inline-block;
        width: 80px;
    }
</style>
<script language="javascript" type="application/javascript">
    $(document).ready(function(){
        $('#btnGetWeather').click(function() {
            $("#myDiv").load("weather.html");
        });
    });
</script>
</head>
<body>
    <center>
        <h2>$.load()调用本服务器内的天气预报</h2>
        <button type="button" id="btnGetWeather">查询</button>
        <hr />
        <div id="myDiv"></div>
    </center>
</body>
</html>
```

程序的运行效果如图 7-20 所示，注意运行时需要把页面部署到 Tomcat 或者 IIS 中。

在本例中的 load 函数的 url 参数是天气预报网页的地址，它将整个网页都载入显示页面中。如果只需要将一部分的内容载入本页面中，可以在 url 中指定需要载入的标签。在本例中如果只想载入 48 小时内的天气预报，只需要将名为 FortyEightHours 的 Div 进行载入，所以 load 函数需做如下的修改。

```
<script language="javascript" type="application/javascript">
    $(document).ready(function(){
        $('#btnGetWeather').click(function() {
```

```
        $("#myDiv").load("weather.html #FortyEightHours");
    });
});
</script>
```

代码修改后的运行效果如图 7-21 所示。



图 7-21 用 load 函数调取本站的资源

3. $.get()函数

当确定以 GET 方式向服务器发起 ajax 请求时，可以直接使用 jQuery 的实用工具函数 $.get()。$.get()函数的语法如下：

```
$.get(url,parameters,callback,dataType)
```

其中，url、parameters、callback 与 $.load()函数的对应参数的作用是一样的。dataType 是预期的服务器响应的数据类型，默认情况下 jQuery 会进行智能判断，该项也为可选项。以下代码将 $.ajax()函数调用的天气预报的 Web Service 改为 $.get()的形式，如例 7-10 所示。

【例 7-10】 使用 $ get ()函数取得天气预报。

```
$.get("http://wthrcdn.etouch.cn/weather_mini",
    {city:city},
    function(data){
        var jsonValue=data;
        jsonValue.data.forecast.forEach(function(e) {
        temp=temp+"<span>"+jsonValue.data.city+
            "</span><span>"+e.date+
            "</span><span>"+e.fengli+
            "</span><span>"+e.fengxiang+
            "</span><span>"+e.high+
            "</span><span>"+e.low+
            "</span><span>"+e.type+
            "</span><br />";
        });
        $("#myDiv").html(temp);
    },
    "jsonp"
)
```

4. $. post()函数

当确定以 POST 方式向服务器发起 ajax 请求时,可以直接使用 jQuery 的实用工具函数 $. post()。 $. post()函数的语法如下:

```
$.post(url,parameters,callback,dataType)
```

$. post()函数中参数的定义与 $. get()函数一样。以下代码将调用的天气预报的 Web Service 改为 $. post()的形式,如例 7-11 所示。

【例 7-11】 使用 $ post()函数取得天气预报。

```
$.post("http://wthrcdn.etouch.cn/weather_mini",
    {city:city},
    function(data){
        var jsonValue=data;
        jsonValue.data.forecast.forEach(function(e) {
        temp=temp+"<span>"+jsonValue.data.city+
            "</span><span>"+e.date+
            "</span><span>"+e.fengli+
            "</span><span>"+e.fengxiang+
            "</span><span>"+e.high+
            "</span><span>"+e.low+
            "</span><span>"+e.type+"</span><br />";
        });
        $("#myDiv").html(temp);
    },
    "jsonp"
)
```

5. $. getJSON()函数

当确定服务器响应的格式为 JSON 格式的时候,可以直接使用 jQuery 的实用工具函数 $. getJSON()。 $. getJSON()函数的语法如下:

```
$.getJSON(url,parameters,callback)
```

$. getJSON()函数的参数的定义与 $. get()函数、$. post()函数一样。在已经判定了服务器返回的数据格式为 JSON 的情况下,不用再设定 dataType 的值。以下代码将调用的天气预报的 Web Service 改为 $. getJSON()的形式,如例 7-12 所示。

【例 7-12】 使用 $ getJSON ()函数取得天气预报。

```
$.getJSON("http://wthrcdn.etouch.cn/weather_mini",
    {city:city},
    function(data){
        var jsonValue=data;
        jsonValue.data.forecast.forEach(function(e) {
        temp=temp+"<span>"+jsonValue.data.city+
            "</span><span>"+e.date+
            "</span><span>"+e.fengli+
```

```
            "</span><span>"+e.fengxiang+
            "</span><span>"+e.high+
            "</span><span>"+e.low+
            "</span><span>"+e.type+"</span><br />";
        });
        $("#myDiv").html(temp);
    }
)
```

6. \$.getScript()函数

通常在编写 HTML 文件时，习惯把全部 JavaScript 文件都写入 Head 中，这样页面在初次加载时就取得了所需要的全部 JavaScript 文件，这样做可能会拖慢页面的加载速度，有些情况下也是没有必要的。虽然可以在需要哪个 JavaScript 文件时，通过 jQuery 动态的创建<script>标签后加入 Head 中，但是这种方式并不理想。为此，jQuery 提供了 \$.getScript()函数来加载 JavaScript 文件，这种方式与加载一个 HTML 片段一样简单方便，并且不需要对 JavaScript 文件进行处理，JavaScript 文件会自动执行。

将 \$.getJSON 的代码作为例子进行改造，具体如例 7-13 所示。

【例 7-13】 使用\$ getScript ()函数取得天气预报。

将取得天气预报的函数独立到 weather.js 的文件中。

```
function getweather() {
    var temp="";
    var city=$("#city").val();

    $.getJSON("http://wthrcdn.etouch.cn/weather_mini",
        {city : city},
        function(data) {
            var jsonValue=data;
            jsonValue.data.forecast.forEach(function(e) {
            temp=temp+"<span>"+jsonValue.data.city+"</span><span>"+
                e.date+"</span><span>"+e.fengli+"</span><span>"+
                e.fengxiang+"</span><span>"+e.high+
                "</span><span>"+e.low+"</span><span>"+e.type+
                "</span><br />";
            });
            $("#myDiv").html(temp);
    })
}
```

调用的这个函数的 jQuery 代码如下所示。

```
$(document).ready(function(){
        $('#btnGetWeather').click(function() {
            $.getScript('js/weather.js',function (){
                getweather();
```

```
            });
        });
});
```

7．关于 JSON 和 JSONP

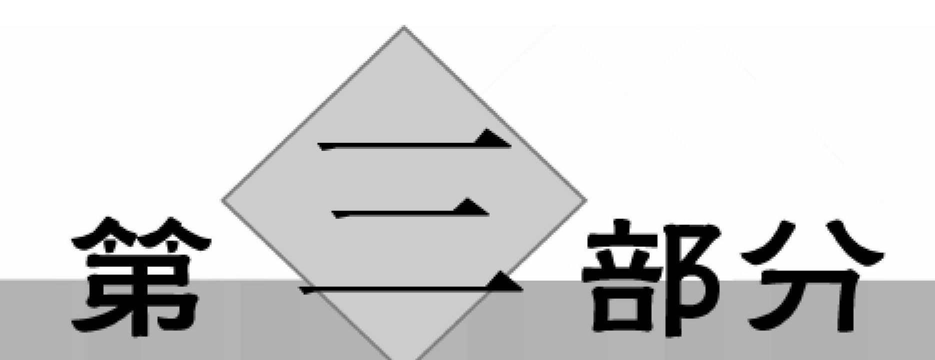
JSON 和 JSONP(JSON with padding)虽然只有一个字母的差别，但其实它们完全就是不同的两个事物，JSON 是一种 ajax 的数据交换格式，而 JSONP 是一种 ajax 用来解决跨域问题的非官方跨域数据交互协议。关于 JSON 在之前的章节中已经做过详细的说明，这里只说明一下 JSONP。

由于同源策略中规定只允许当前源（域名、协议、端口）的资源进行通信，也就是说位于不同的源下的 ajax 请求不能取得对方的数据。不过在 Web 页面上调用 JavaScript 文件时，则不受是否跨域的影响。为了实现跨域请求，一种可行的方案是在远程服务器上设法把数据装进 JavaScript 格式的文件里，然后在请求端通过 HTML 的＜script＞标签实现跨域请求装有数据的 JavaScript 文件，从而解决了跨域的数据请求。而 JSON 的纯字符数据格式可以简洁地描述复杂数据，且被 JavaScript 支持，在客户端几乎可以随心所欲地处理这种格式的数据的特性让 JSON 成为跨域传输的理想格式。所以到目前为止最被推崇或者说首选的方案是用 JSON 来传输数据，靠 JSONP 来实现跨域。

本章小结

本章对 Web 开发中的客户端数据请求技术进行了详细的讲解，着重介绍了 Web Server、HTTP 请求，以及 ajax 技术，这三种技术是目前在 Web 开发中进行前后台交互的主流技术。每种技术在讲解的时候，都遵循由浅入深的原则，先讲解其基础技术，再在其上进行扩展说明，最后对这种技术进行讲解。在讲解的过程中每个重要的知识点都配有典型的案例，可以很好地帮助学习者理解及掌握。





实　战　篇



第 8 章　在线订餐网站

8.1　项目背景

随着电子商务的快速发展，人们已经习惯于在互联网上选购自己所需的各种商品。但是对于餐饮行业的各种菜品来说，它无法和传统商品一样适合长距离的运输。为了保持菜品的新鲜和美味，必须要在很短的时间内就送到顾客的手中。快速的配送成为网上订餐能否成功的关键条件。在网上订餐还没有发展起来之前，人们普遍采取电话订餐的方式，但是这样的方式有明显的弊端，比如菜品展示不直观；商家推广困难，只能靠顾客间相互推荐；商家需要请专门的员工来配送，使得成本增加等。伴随着近几年餐饮外卖的配送业务的发展，出现了美团、百度等大型外卖配送公司，解决了配送问题之后，促进了网上订餐的大发展，这也让人们更喜欢在网上订餐，人们在足不出户的情况下，就能享受到美味的食物。

本章利用所学的知识来开发一个简易的网上订餐系统，要求系统能提供良好的用户界面和良好的用户体验感是本系统设计的目标。系统功能需满足以下需求。

（1）用户能够随意浏览菜品，并能获取菜品的详细信息，如菜品图片预览、菜品介绍，以及用户的评价。

（2）用户在下订单之前必须要成为注册用户，且可以对自己的个人信息进行维护。

（3）注册用户可以管理自己的购物车以及订单。

（4）系统要提供餐厅后台管理端程序，管理员需要登录后才能进入管理程序。

（5）管理员可以在后台对菜品信息进行维护。

（6）管理员可以在后台进行订单处理。

8.2　系统需求和设计

根据软件开发的流程，在完成了用户需求调研之后，需要进行程序的设计工作，为简约起见，本系统只进行功能设计、数据库设计和程序设计三个部分。

8.2.1　功能设计

根据系统的需求描述，得到如图 8-1 所示的功能模块关联图。

8.2.2　数据库设计

根据需求分析和概要设计的结果，本系统需要设计完成以下的数据库表。

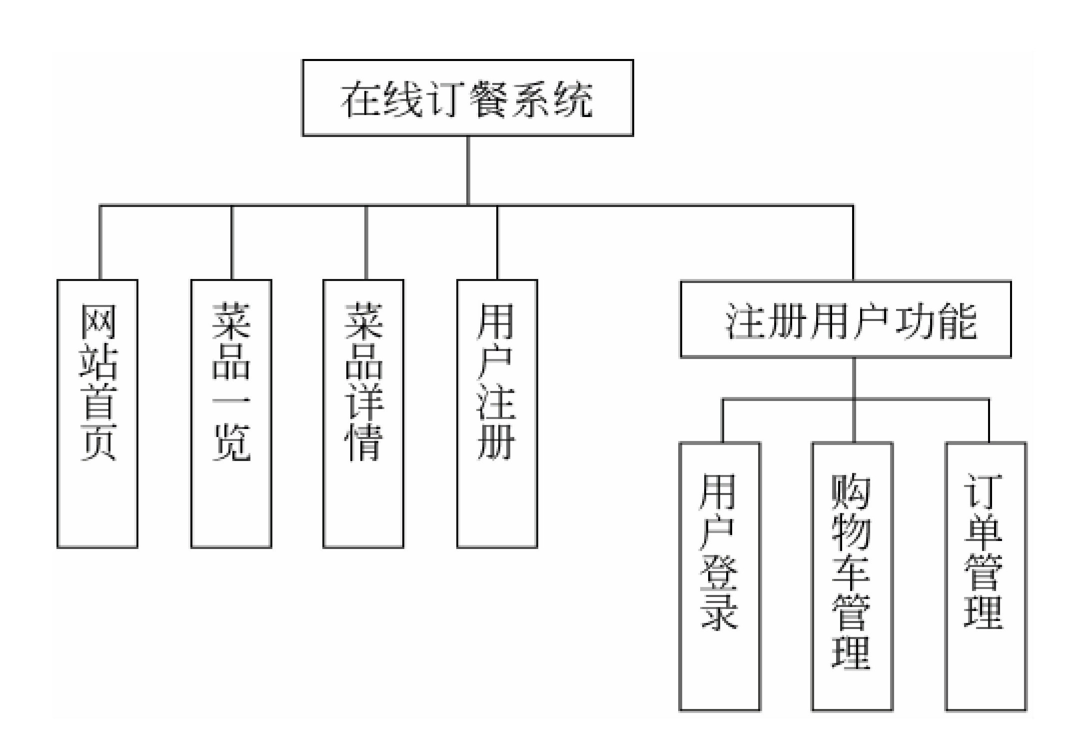


图 8-1　功能模块关联图

（1）菜品表(food)：用于保存菜品相关的信息，该表是整个系统的基础表，所有业务处理中需要用到的菜品信息都需要从本表中取得。具体的结构如表 8-1 所示。

表 8-1　菜品表(food)

序号	字 段 名	中文名称	类　型	长度	主键	备　　注
1	fID	菜品 ID	varchar	50	是	
2	fName	菜品名称	varchar	100		
3	fDescription	菜品描述	varchar	100		
4	fType	菜品类型	varchar	8		中餐、西餐等
5	fHobbies	菜品口味	varchar	50		
6	fTechnology	工艺	varchar	50		煎、炒、炖、焖、烤、炸、蒸等
7	fSeasonal	季节推荐	varchar	50		春、夏、秋、冬
8	fPrice	价格	double			
9	fImage	菜品图片	varchar	100		图片地址
10	fSalesVolume	销量	Int			

（2）点评表(comment)：用于保存客户对菜品的评价信息，用户在购买之前或者购买之后的点评都需要记录在本表中。具体的结构如表 8-2 所示。

表 8-2　点评表(comment)

序号	字 段 名	中文名称	类　型	长度	主键	备　　注
1	cID	点评 ID	varchar	50	是	
2	fID	菜品 ID	varchar	50		关联菜品表
3	buyerName	用户账号	varchar	12		关联用户表
4	evaluate	评价内容	varchar	500		
5	time	点评时间	datetime			

（3）购物车表(shop_cart)：用于保存注册用户的购物车信息。具体的结构如表 8-3 所示。

表 8-3　购物车表(shop_cart)

序号	字 段 名	中文名称	类　型	长度	主键	备　　注
1	fID	菜品 ID	varchar	50		关联菜品表
2	buyerName	用户账号	varchar	12		关联用户表

(4) 订单表(order):用于保存注册用户的菜品订单信息,具体的结构如表 8-4 所示。

表 8-4　订单表(order)

序号	字 段 名	中文名称	类　型	长度	主键	备　　注
1	orderID	订单 ID	varchar	50	是	
2	fID	菜品 ID	varchar	50		关联菜品表
3	buyerName	用户账号	varchar	12		关联用户表
4	quantity	购买数量	int			默认值为 1
5	pay	支付金额	double			
6	tradingStatus	交易状态	varchar	12		
7	deliveryStatus	发货状态	varchar	12		
8	time	下单时间	datetime			

(5) 用户表(buyer):用于保存注册用户的个人相关信息,具体的结构如表 8-5 所示。

表 8-5　用户表(buyer)

序号	字 段 名	中文名称	类　型	长度	主键	备　　注
1	buyerName	用户账号	varchar	12	是	
2	password	密码	varchar	12		
3	receiver	收货人	varchar	100		
4	Address	收货地址	varchar	100		
5	Tel	收货人电话	varchar	11		

8.2.3　程序设计

一套好的系统离不开良好的架构设计。由于本项目的特殊性,需要兼容 Java 和 C# 语言两种后台语言,所以在浏览器端只能采取静态 HTML 和 jQuery,浏览器端和服务器端的通信采用之前讲过的 jQuery 的 ajax 来完成。程序的整体结构如图 8-2 所示。

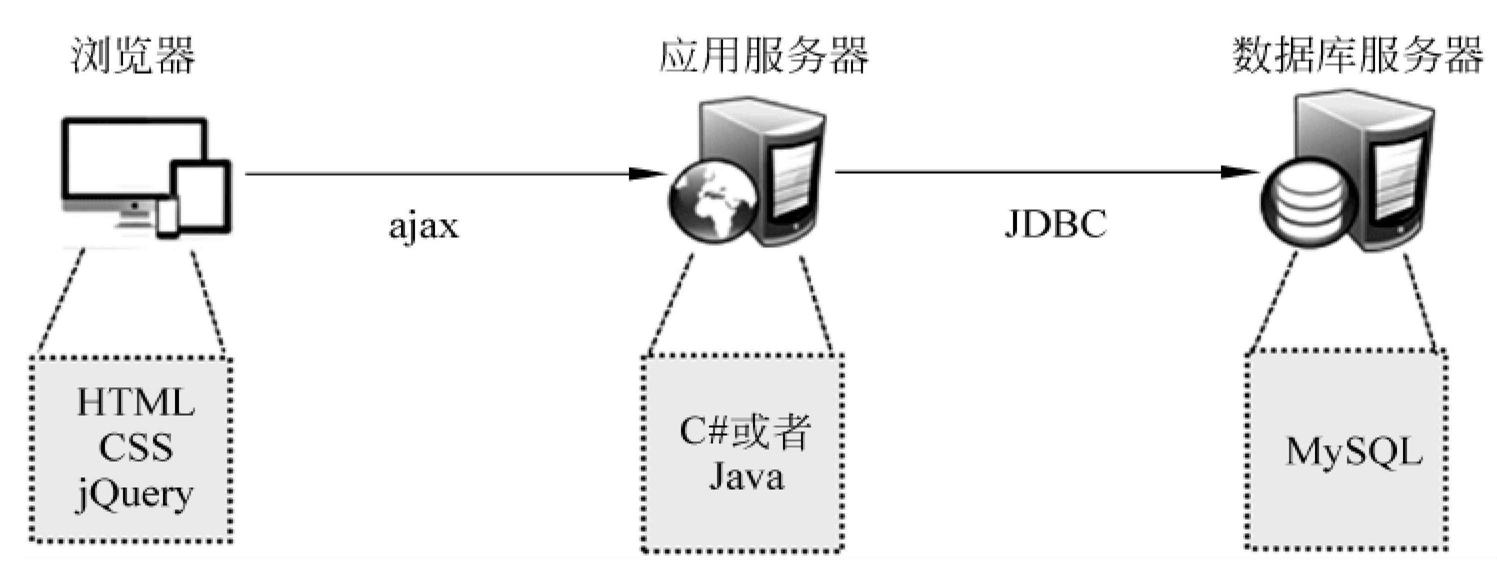


图 8-2　程序的整体结构

对于在应用服务器部分的代码，为了能对 C# 和 Java 都做出较好的约束，也为了能共用浏览器端的全部代码，此处采用了工厂方法模式来控制业务逻辑的访问方式。示意图如图 8-3 所示。

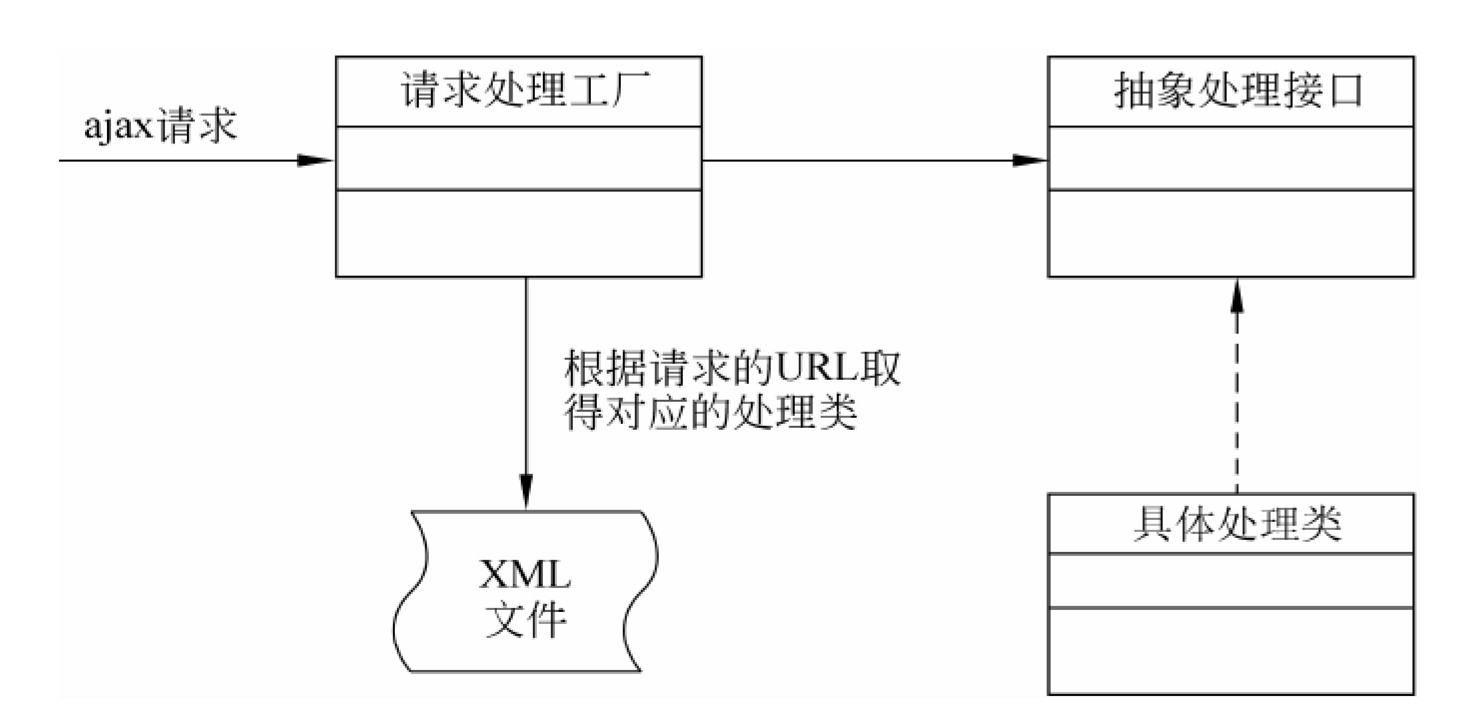


图 8-3　应用服务器处理请求的程序结构

基于 XML 文件的平台无关性，在该文件中定义的 ajax 请求和处理程序的对应关系，可以在 Java 和 C# 语言中通用。该文件(RequestOperation.xml)的内容如下所示。

```
<?xml version="1.0" encoding="utf-8" ?>
<config>
    <request id="Login">Action.Login</request>
    <request id="Register">Action.Register</request>
    <request id="ShopCart">Action.ShopCart</request>
    <request id="AddInCart">Action.AddInCart</request>
    <request id="DeleteInCart">Action.DeleteInCart</request>
    <request id="DeleteOrder">Action.DeleteOrder</request>
    <request id="Comment">Action.Comment</request>
    <request id="AddComment">AddComment</request>
    <request id="Order">Action.Order</request>
    <request id="FoodList">Action.FoodList</request>
    <request id="FoodDetail">Action.FoodDetail</request>
</config>
```

该文件起到的另外一个作用是规定了对应处理类的名称，也就是说不管是在 Java 中还是在 C# 中类名都必须保持一致。

(1) Java 中的共通代码。在 Java 中，需要借助于 Servlet 来完成请求的处理，所以在 Java 中就需要建立一个 Servlet 来充当请求处理工厂类，用它来处理所有的 ajax 请求。在 Eclipse 的工程中创建一个名为 ajaxQuestServletFactory 的 Servlet，并把 web.xml 文件中的配置文件修改为以下所示的代码。

```
<servlet>
    <description></description>
    <display-name>ajaxQuestServletFactory</display-name>
    <servlet-name>ajaxQuestServletFactory</servlet-name>
```

```
    <servlet-class>ajaxQuestServletFactory.ajaxQuestServletFactory
    </servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>ajaxQuestServletFactory</servlet-name>
    <url-pattern>/*</url-pattern>
</servlet-mapping>
```

倒数第二行的<url-pattern>节点配置了所有的请求，都由 ajaxQuestServletFactory 类来处理。ajaxQuestServletFactory.java 的核心代码如下所示。

```
/**
 * 这是控制 Servlet 访问的工厂
 */
public class ajaxQuestServletFactory extends HttpServlet {
    private static final long serialVersionUID=1L;

    public ajaxQuestServletFactory() {
        super();
    }

     protected void doGet (HttpServletRequest request, HttpServletResponse
     response)
            throws ServletException, IOException {
        //请求信息格式化
        request.setCharacterEncoding("utf-8");
        //设置响应头信息
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-store");//Http1.1
        response.setHeader("Pragma", "no-cache");//Http1.0
        response.setDateHeader("Expires", 0);
        //取得请求的参数
        String xmlId=request.getParameter("ajax");
        try{
            //创建 document 对象
            DocumentBuilderFactory factory=DocumentBuilderFactory.
            newInstance();
            DocumentBuilder builder=factory.newDocumentBuilder();
            //获取 XML 文件的地址
            String urlpath=ajaxQuestServletFactory.class.
                getResource("RequestOperation.xml").toURI().getPath();
            //取得 XML 文件,返回给 document 对象
            Document document=builder.parse(new File(urlpath));
            //获取根节点元素
            Element root=document.getDocumentElement();
            NodeList node=root.getElementsByTagName("request");
            //循环读取 XML 文件的节点并与请求的 ajax 对比,找到对应的具体处理类
            for(int i=0;i<node.getLength();i++){
                Element xmlRequest=(Element) node.item(i);
```

```
                //获取 request 标签的 id 属性值
                String id=xmlRequest.getAttribute("id");
                if(id.equals(xmlId)){
                    //生成对应的 action
                    Action action=null;
                    action=(Action)Class.forName(xmlRequest.getTextContent()).
                    newInstance();
                    action.doGets(request, response);
                }
            }
        } catch (ParserConfigurationException e) {
        //找不到具体处理类时即抛出异常
            System.out.println("无法获取请求对象");
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
                          throws ServletException, IOException {
        //将 post 请求也转到 get 请求中处理
        doGet(request, response);
    }
}
```

Action 作为接口里面定义了接收处理的方法，具体的代码如下所示。

```
public interface Action {
    public abstract void handle(HttpServletRequest request, HttpServletResponse
    response) throws IOException;
}
```

为了在程序的各个地方能方便地使用数据库连接，需要对数据库连接创建统一的类来管理对数据库的访问，核心代码如下所示。

```
/**
 * MySQL 数据库连接类
 *
 * /
public class DB {
    private Connection conn=null;
    private Statement stmt=null;
    private ResultSet rst=null;

    public DB(){
        String drivaerName="com.mysql.jdbc.Driver";    //数据库驱动程序的名称
        try {
            Class.forName(drivaerName);
        } catch (ClassNotFoundException e) {
```

```
            //TODO Auto-generated catch block
            System.out.println("数据库驱动程序加载失败");
        }
    }

    /**
     * 打开数据库
     * @return conn
     * /
    public Connection getConn(){
        //MySQL 的数据库连接串
        String url= "jdbc:mysql://115.159.192.89:3306/sellingwebsite?
        characterEncoding=utf-8";
        String user="root";
        String upassword="x5";
        try {
            conn=DriverManager.getConnection(url,user,upassword);
        } catch (SQLException e) {
            //TODO Auto-generated catch block
            System.out.println("打开数据库失败");
            e.printStackTrace();
        }
        return conn;
    }

    /**
     * 生成一个发送 SQL 语句的 Statement
     * @return stmt
     * /
    public Statement getStmt(){
        try {

            conn=getConn();
            stmt=conn.createStatement();
            //stmt=conn.createStatement();
        } catch (SQLException e) {
            //TODO Auto-generated catch block
            e.printStackTrace();
        }
        return stmt;
    }

    /**
     * 执行增、删、改的 SQL 命令
     * @param SQL
     * @return r
     * /
    public int executeUpdate(String sql){
        int r=0;
```

```
        try {
            conn=getConn();
            stmt=getStmt();
            r=stmt.executeUpdate(sql);
        } catch (SQLException e) {
            //TODO Auto-generated catch block
            System.out.println(sql);
            System.out.println("执行增、删、改的 SQL 命令失败");
        }
        return r;
    }

    /**
     * 执行查询的 SQL 命令
     * @param SQL
     * @return rs
     * /
    public ResultSet executeQuery(String sql){
        try {
            conn=getConn();
            stmt=getStmt();
            rst=stmt.executeQuery(sql);
        } catch (SQLException e) {
            //TODO Auto-generated catch block
            System.out.println(sql);
            System.out.println("执行查询 SQL 的命令失败");
        }
        return rst;
    }

    /**
     * 关闭数据库并释放资源
     * /
    public void closeDB(){
        try {
            if(rst!=null){
                rst.close();
            }
            if(stmt!=null){
                stmt.close();
            }
            if(conn!=null){
                conn.close();
            }
        } catch (SQLException e) {
            //TODO Auto-generated catch block
            System.out.println("关闭数据失败");
        }
    }
}
```

（2）C# 中的共通代码

在 C# 中可以创建 HomeController.cs 类来处理所有的 ajax 请求，核心代码参考如下：

```
public class HomeController : Controller
    {
        [HttpPost]
        public new ActionResult Request(FormCollection collection, string ajax)
        {
            collection.Remove("ajax");
            //XML 文件的操作
            XmlDocument xml=new XmlDocument();
            xml.Load(Server.MapPath("~/RequestOperation.xml"));
            XmlNodeList xnd=xml.SelectNodes("config/request");
            string ajaxType="";
            //循环 XML 文件的节点，找到和请求一致的参数节点
            foreach (XmlNode item in xnd)
            {
                if (item.Attributes[0].Value==ajax)
                {
                    ajaxType=item.Attributes[0].Value;
                    break;
                }
            }
            //ajax 类型查找不到的情况下会返回异常
            if (string.IsNullOrEmpty(ajaxType))
            {
                return HttpNotFound();
            }
            //利用反射创建具体的调用类
            ajaxType="DAL.Factory."+ajaxType;
            IDAL obj;
            lock(this)
            {
                obj=(IDAL)Assembly.Load("DAL").CreateInstance(ajaxType);
            }
            return Json(obj.Execute(collection));
        }
    }
```

创建 IDAL.cs，充当抽象产品角色，具体代码如下：

```
public interface IDAL
    {
        IJson Execute(FormCollection form);
    }
```

共通的数据库访问代码类 SqlHelper.cs 的代码如下：

```
public class SqlHelper
{
    private static MySqlConnection con=new MySqlConnection();
    private static MySqlCommand cmd=new MySqlCommand();

    //打开数据库连接
    private static void OpenConnection()
    {
        if (con.State==System.Data.ConnectionState.Closed)
        {
            string conString= "server=115.159.192.89;uid=root;pwd=x5;
            database=sellingwebsite";
            con.ConnectionString=conString;
            cmd.Connection=con;
            con.Open();
        }
    }

    //关闭数据库连接
    private static void CloseConnection()
    {
        if (con.State==System.Data.ConnectionState.Open)
        {
            con.Close();
        }
    }

    //执行无参数的非查询语句
    public static int GetExecuteNonQuery(string sqlString)
    {
        OpenConnection();
        cmd.CommandType=System.Data.CommandType.Text;
        cmd.CommandText=sqlString;
        int result=cmd.ExecuteNonQuery();
        CloseConnection();
        return result;
    }

    //执行带参数的非查询语句
    public static int GetExecuteNonQuery(string sqlString,params
    MySqlParameter[] values)
    {
        OpenConnection();
        cmd.CommandType=System.Data.CommandType.Text;
        cmd.CommandText=sqlString;
        cmd.Parameters.Clear();
        cmd.Parameters.AddRange(values);
        int result=cmd.ExecuteNonQuery();
        cmd.Parameters.Clear();
        CloseConnection();
```

```
        return result;
    }

    //执行无参数的查询语句
    public static object GetExecuteScalar(string sqlString)
    {
        OpenConnection();
        cmd.CommandType=System.Data.CommandType.Text;
        cmd.CommandText=sqlString;
        object result=cmd.ExecuteScalar();
        CloseConnection();
        return result;
    }

    //执行带参数的查询语句
    public static object GetExecuteScalar(string sqlString, params
    MySqlParameter[] values)
    {
        OpenConnection();
        cmd.CommandType=System.Data.CommandType.Text;
        cmd.CommandText=sqlString;
        cmd.Parameters.Clear();
        cmd.Parameters.AddRange(values);
        object result=cmd.ExecuteScalar();
        CloseConnection();
        cmd.Parameters.Clear();
        return result;
    }

    public static MySqlDataReader GetExecuteReader(string sqlString)
    {
        OpenConnection();
        cmd.CommandType=System.Data.CommandType.Text;
        cmd.CommandText=sqlString;
        MySqlDataReader reader=cmd.ExecuteReader();
        return reader;
    }

    public static MySqlDataReader GetExecuteReader(string sqlString,
    params MySqlParameter[] values)
    {
        OpenConnection();
        cmd.CommandType=System.Data.CommandType.Text;
        cmd.CommandText=sqlString;
        cmd.Parameters.Clear();
        cmd.Parameters.AddRange(values);
        MySqlDataReader reader=cmd.ExecuteReader();
        cmd.Parameters.Clear();
        return reader;
    }
}
```

8.3 功能实现

在前面的章节中对系统的业务以及程序的结构设计进行了说明，这些工作为系统的具体实现提供了充分的准备，在本节中我们将展示系统的一些实现效果以及核心功能的具体代码。

8.3.1 首页

订餐网站的首页是网站的门户，所以一个美观大方的首页能很好地吸引用户，它直接影响到用户对订餐网站的第一印象。

(1) 网页设计。按照网页设计的规则，在首页中应该包含网站的导航栏，以及热门菜品的展示功能。整体的设计界面如图 8-4 所示。



图 8-4 网站首页

（2）导航栏的实现。导航栏中有首页、菜单、账户管理、购物车、注销 5 个一级菜单，单击相应的菜单能进入对应的界面。特别是在购物车菜单上设置了当鼠标悬停时能展开购物车内物品清单的功能，且提供了立即支付的功能。导航栏的具体 HTML 实现如以下代码所示。

```
<nav class="nav wow fadeInDown animated">
    <div class="container">
        <ul class="navbar" style="z-index:99;position: absolute">
            <li class="active"><a href="index.html">首页</a></li>
            <li><a href="menu.html">菜单</a></li>
            <li class="animated fadeInDown" data-title="true"
                style="width: 383px;text-align: center;font-weight:
                bold;font-size: 26px;">Online Restaurant</li>
            <li class="userstate">
            </li>
            <li class="carBtn">
                <a href="#">购物车</a>
            </li>
            <div class="shopCar animated"
                style="animation-duration:0.5s;position:absolute;width: 240px;
                  top:50px;right: 0px;height: 0px;float: right;display: none">
                <ul>
                    <li>购物车</li>
                    <li class="CarFoodList">
                        <div>
                            <ul class="CarList" id="CarList">
                                <p id="CarProcess" style="display: none;
                                    background-color: #00f0ff;width: 0%;height:
                                    20px;margin: 60px 0;">
                                </p>
                            </ul>
                        </div>
                        <script type="text/javascript">
                            $(function () {
                                $("#CarProcess").animate({width:"90%"},2000)
                                var LoadCar=self.setInterval(function ()
                                {$("#CarProcess").animate({width:"0%"},500).
                                animate({width:"90%"},2000)
                                },2000)
                            })
                        </script>
                    </li>
                </ul>
                <a href="#" id="btnPayfor">立即支付</a>
            </div>
            <div class="searchPanel" style="height: 50px;width: 48px;float:
                right;
                position: relative">
```

```
                <div class="SignOut animated fadeInRight" style="display:
                block;">
                    <div id="SignOut_Up" class="Up_Back">
                        <input type="button" id="SignOut" value="注销"
                            style="font-size:12px;line-height:30px;text-
                            align:center;
                            border:0;outline:0;width:100%;
                            background:Transparent;display:block">
                    </div>
                </div>
            </div>
        </ul>
        <div class="modalback"
            style="z-index: 0;position: absolute;top:0px;background-color:
                #FFF;
                height: 50px;width: 1200px;opacity: 0.2">
        </div>
    </div>
</nav>
```

另外，在页面 Head 部分还需引入对应的 CSS 文件以及 JavaScript 文件，代码如下所示。

```
<head>
    <meta charset="UTF-8">
    <link href="css/reset.css" rel="stylesheet" type="text/css" />
    <link href="css/style.css" rel="stylesheet" type="text/css" />
    <link href="css/myanimation.css" rel="stylesheet" type="text/css" />
    <script src="js/jquery-2.2.4.min.js"></script>
    <script src="js/jquery.cookie.js"></script>
    <script src="js/formSubmit.js"></script>
    <script src="js/myapp.js"></script>
    <script src="js/wow.min.js"></script>
    <script>
        new WOW().init();
    </script>
    <title>Index</title>
</head>
```

CSS 文件中的 myanimation. css 和 JavaScript 中的 jquery-2. 2. 4. min. js、jquery. cookie. js 及其他框架的代码请参考最终程序。导航栏的 CSS 代码如下所示。

```
.nav {
    width: 100%;
    min-height: 50px;
    position: relative;
    z-index: 99;
}
.container {
    width: 1200px;
```

```
    margin: 0 auto;
    min-height: 50px;
    line-height: 50px;
    background: rgba(255, 255, 255, 0.45098);
}
.container>.navbar {
    width: 1200px;
}
.container>.navbar>li {
    width: 191px;
    float: left;
    border-right: 1px solid #FFF;
}
.hrefLogin {
    width: 95px;
    float: left;
}
.iLogin {
    width: 96px;
    float: left;
}
.iLogin:hover {
    cursor: pointer;
}
.container>.navbar>li a {
    font-family: "Roboto Slab", serif;
    font-size: 18px;
    font-weight: 600;
    display: block;
    text-align: center;
}
```

（3）明星菜品部分的实现。主页中需要展示主要的菜品信息，为了实现该部分的功能，本例设置了背景图片等信息，用来衬托菜品。该部分的具体实现如下所示。

```
<div class="banner">
    <div class="content">
        <h3 class="InLeftBig animated wow"
            data-wow-duration="1000ms"
            data-wow-delay="300ms"
            style="font-size: 4em;margin: 0;color: #fff;margin-top: 50px;">
            Welcome to the Online Restaurant
        </h3>
        <ul class="slidefood animated wow fadeInDown" data-wow-duration="1000ms"
        data-wow-delay="300ms" style="margin-top: 20px">
            <li>
                <div class="banner-info1-grid">
                    <a href="detail.html" style="display: inline-block">
```

```
                <img src="img/food1.jpg" alt=" " draggable="false"
                width="250"
                    style="border-radius: 50%;border: 2px solid #FFF" />
            </a>
            <div class="title-backround">
                <h3>水煮鱼</h3>
                <p>满目的辣椒红亮养眼,辣而不燥,麻而不苦。"麻上头,辣过瘾",
                    让水煮鱼在全国流行得一塌糊涂。</p>
            </div>
        </div>
    </li>
    <li>
        <div class="banner-info1-grid">
            <a href="detail.html" style="display: inline-block">
                <img src="img/food2.jpg" alt=" " draggable="false"
                    width="250"
                    style="border-radius: 50%;border: 2px solid #FFF" />
            </a>
            <div class="title-backround">
                <h3>牛排</h3>
                <p>牛肉是王公贵族们的高级肉品,尊贵的牛肉被他们搭配上了当时
                    也是享有尊贵身份的胡椒及香辛料等一起烹调,并在特殊场合中
                    供应,以彰显主人的尊贵身份。</p>
            </div>
        </div>
    </li>
    <li>
        <div class="banner-info1-grid">
            <a href="detail.html" style="display: inline-block">
                <img src="img/food3.jpg" alt=" " draggable="false"
                    width="250"
                    style="border-radius: 50%;border: 2px solid #FFF" />
            </a>
            <div class="title-backround">
                <h3>意大利面</h3>
                <p>作为意大利面的法定原料,杜兰小麦是最硬质的小麦品种,具有
                    高密度、高蛋白质、高筋度等特点,其面紧实又有弹性,并且会根
                    据不同面酱而决定口感</p>
            </div>
        </div>
    </li>
    <li>
        <div class="banner-info1-grid">
            <a href="detail.html" style="display: inline-block">
                <img src="img/food4.jpg" alt=" " draggable="false"
                width="250"
                    style="border-radius: 50%;border: 2px solid #FFF" />
            </a>
            <div class="title-backround">
                <h3>白灼虾</h3>
```

```
                    <p>烹饪工艺是白灼。"白灼"两字指的是将原汁原味的鲜虾直接放进
                       清水里煮食。广州人喜欢用白灼之法来做虾,为的是保持其鲜、
                       甜、嫩的原味,然后将虾剥壳并蘸酱汁而食。</p>
                </div>
            </div>
        </li>
    </ul>

    <h1 class="fadeInUp wow animated"
        style="width: 300px;background:rgba(246, 90, 91, 0.66);
            margin-top: 10px;margin-left:auto;margin-right:auto;border-
            radius: 5px;"
        data-wow-duration="1000ms" data-wow-delay="300ms">
        <a href="menu.html" class="readMorebtn" style="color: #FFF;
            display: block;padding: 12px 0">Read More</a>
    </h1>

    <div class="FoodClassify flipInX animated wow"
            data-wow-duration="1000ms" data-wow-delay="300ms">
        <div class="ClassifyContainer">
            <ul class="ClassifyMenu">
                <li>
                    <a href="#">
                        <div class="NormalFood">
                            <div class="icon">
                                <img src="img/k1.png" />
                            </div>
                            <h4 class="className">日常菜品</h4>
                            <p class="classContent">
                                以食为天的国度里,饮食礼仪自然成为饮食文化的一个
                                重要部分。家人、朋友聚会,怎么能少了它?
                            </p>
                        </div>
                    </a>
                </li>
                <li>
                    <a href="#">
                        <div class="Drink">
                            <div class="icon">
                                <img src="img/k2.png" />
                            </div>
                            <h4 class="className">茶品</h4>
                            <p class="classContent">
                                忙碌的生活中,是不是应该喝杯下午茶,解解乏。和朋
                                友洽谈的时候,是不是应该边喝茶边闲聊?
                            </p>
                        </div>
                    </a>
                </li>
                <li>
```

```
                        <a href="#">
                            <div class="FastFood">
                                <div class="icon">
                                    <img src="img/k3.png" />
                                </div>
                                <h4 class="className">快餐</h4>
                                <p class="classContent">
                                    生活就应该有生活的样子,工作狂必备,快餐只需一单
                                    解决!
                                </p>
                            </div>
                        </a>
                    </li>
                </ul>
            </div>
        </div>
    </div>
</div>
```

关于主页最后部分的实现,由于都是 HTML 和 CSS 的应用,故在此省略,具体请查询本章参考程序的源代码。

8.3.2 菜品一览和菜品详情

在首页功能完成之后,用户需要查看更多的菜品时,需要到菜品一览界面中去查看。如果需要查看更多的关于菜品的信息,可以单击菜品图片进入菜品详情界面,该页面中展示了更多的菜品信息以及用户对该菜品的评价。

(1) 网页设计。界面保持了和首页一样的风格,具体的界面设计如图 8-5 和图 8-6 所示。

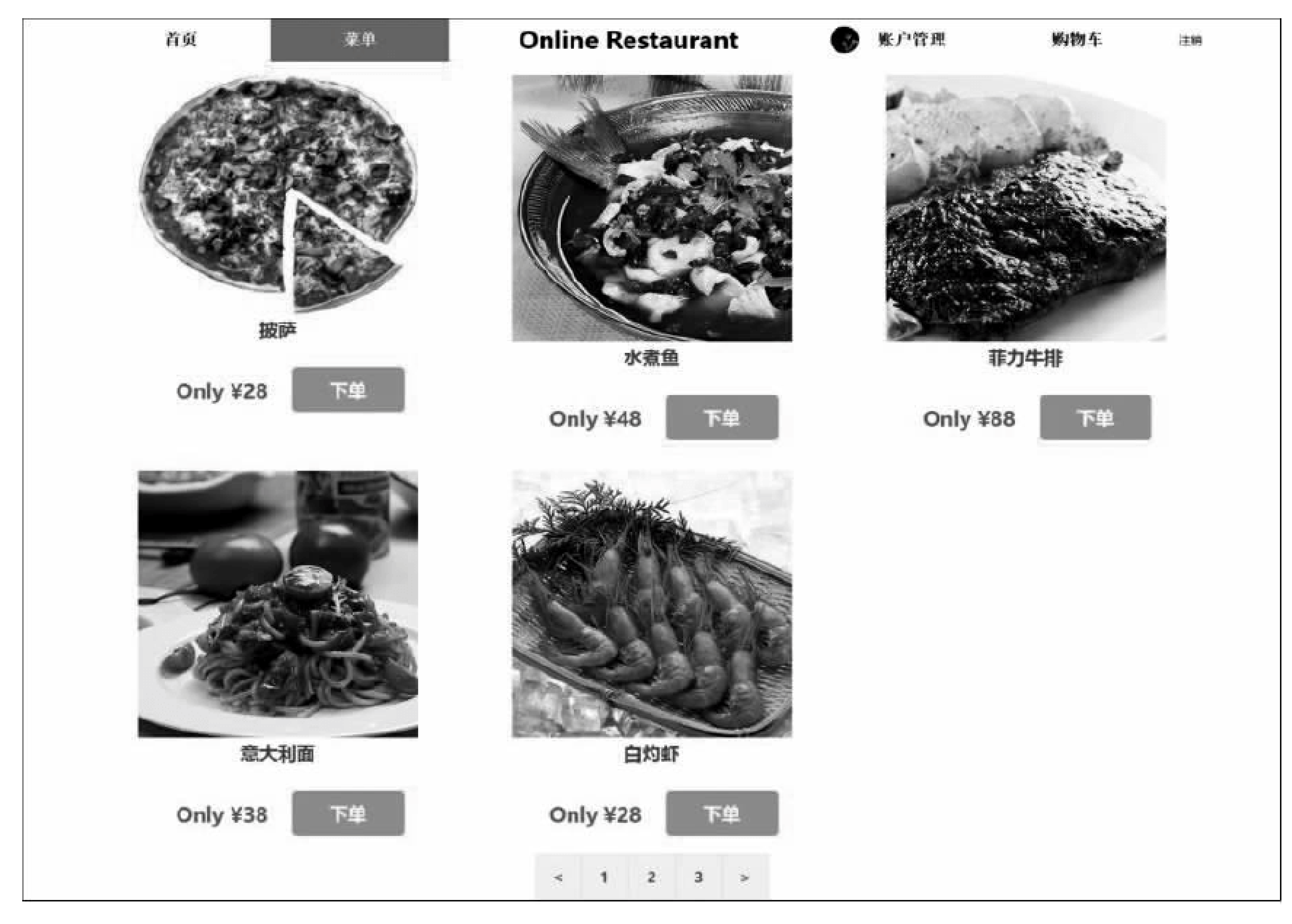


图 8-5 菜品的浏览界面

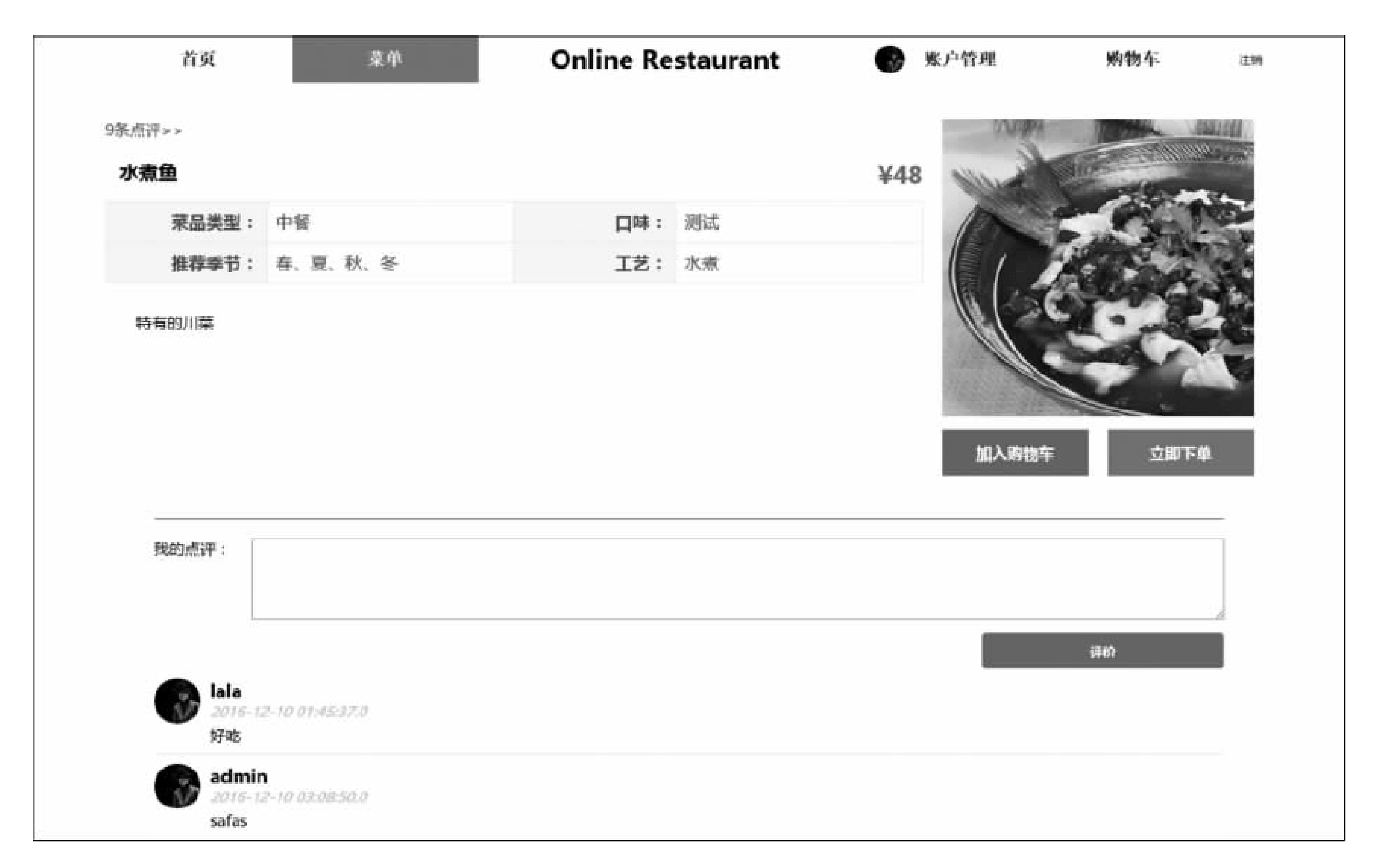


图 8-6　菜品的详情界面

(2) 菜品一览界面的实现。该页面的实现采用以下方法：在页面加载完成时向服务器发送 ajax 请求并取得菜品数据，再以 JSON 的形式返回到浏览器中，拼接成 HTML 语句以用于显示。所有该部分的前台代码比较简单，具体的 HTML 代码如下所示。

```
<div class="FoodList" id="FoodList">
</div>

<div class="pageChange">
    <ul class="page">
        <li><a href="#">&lt;</a></li>
        <li><a href="#">1</a></li>
        <li><a href="#">2</a></li>
        <li><a href="#">3</a></li>
        <li><a href="#">&gt;</a></li>
    </ul>
</div>
```

名为 FoodList 的 DIV 就是用来显示菜品清单的。为了介绍代码，需要将这个 DIV 中的样式信息预先定义，具体代码如下所示。

```
.FoodList {
    width: 1200px;
    margin: 0 auto;
}

.FoodItem {
    float: left;
    text-align: center;
    padding: 15px;
```

```
    width: 350px;
    color: #666;
    margin: 0 10px;
}

.FoodItem img {
    width: 300px;
}

.FoodItemContent {
    text-align: center;
}

.FoodItemContent>h1 {
    font-size: 20px;
    text-align: center;
}

.FoodItemContent>div {
    width: 270px;
    display: inline-block;
}

.FoodItemContent>div>h1:first-of-type {
    width: 150px;
    text-align: center;
    float: left;
    display: inline-block;
    margin: 30px auto;
    color: #5A9522;
    padding: 12px 0;
    font-size: 22px;
}

.FoodItemContent>div>h1:last-of-type {
    font-size: 20px;
    width: 120px;
    text-align: center;
    background: rgba(246, 90, 91, 0.66);
    margin: 30px auto;
    border-radius: 5px;
    float: right;
    display: inline-block;
}

.FoodItemContent>div>h1:last-of-type>a {
    color: #FFF;
    display: block;
    padding: 12px 0;
}
```

关于翻页控制的 CSS 代码如下所示。

```
.pageChange {
    width: 1200px;
    height: 50px;
    margin: 0 auto;
    text-align: center;
    clear: both;
}

ul.page {
    width: 250px;
    height: 50px;
    display: inline-block;
    border-radius: 10px;
}

ul.page>li {
    width: 48px;
    height: 50px;
    line-height: 50px;
    border: 1px solid #eee;
}

ul.page>li a {
    background-color: #f5f5f5;
    display: block;
    color: #777;
    font-weight: bold;
}

ul.page>li a:hover {
    color: #000;
}
```

页面加载时，利用 jQuery 的 ready 函数向服务器发送 HTTP 请求并取得菜单信息的 JavaScript 代码如下所示。

```
<script>
    $(document).ready(function () {
        LoadMenu()
    })

    //加载菜单
    var LoadMenu=function () {
        //发送 HTTP 请求
        var options={
            url: "./ajaxQuestServletFactory",
            type: "POST",
            data: "ajax="+"FoodList",
```

```
            dataType: "Json",
            async: false
        };
        $.ajax(options).done(function (obj) {
            var Foods=obj.FoodList;
            $.each(Foods, function (index, values) {
                //标题
                ElementFName="<h1>"+values.FName+"</h1>";
                //span 价格
                ElementSPrice="<span>&yen;"+values.FPrice+"</span>";
                //DIV 价格
                ElementPrice="<h1>Only "+ElementSPrice+"</h1>";
                ElementBuyA="<a href='detail.html?fid="+values.FId+"'
                              class='readMorebtn'>下单</a>";
                ElementBuyAh1="<h1>"+ElementBuyA+"</h1>"
                ElementPriceBuy="<div>"+ElementPrice+ElementBuyAh1+"</div>";
                ElementContent="<div class='FoodItemContent animated fadeIn'>"+
                               ElementFName+ElementPriceBuy+"</div>";
                ElementImg="<img src='"+values.FImage+"'/>";
                ElementImgA="<a href='detail.html?fid="+values.FId+"'
                              class='animated fadeIn'>"+ElementImg+"</a>";
                ElementFoodItem="<div class='FoodItem'>"+ElementImgA+
                                  ElementContent+"</div>";
                $("#FoodList").append(ElementFoodItem);
            })
        })
        return false;
    }
</script>
```

Web 服务器端的 Java 代码如下所示。

```
/**
 * 加载菜品
 */
public class FoodList implements Action {
      public  void  handle  (HttpServletRequest  request,  HttpServletResponse
      response) throws IOException {
        ArrayList<foodBean>list=new food_Dao().FoodList();
        ArrayList<JSONObject>foodlist=new ArrayList<JSONObject>();
        try {
            for(int i=0; i<list.size();i++){
                JSONObject foodJson=new JSONObject();

                foodJson.put("FId", list.get(i).getfID());
                foodJson.put("FName", list.get(i).getfName());
                foodJson.put("FPrice", list.get(i).getfPrice());
                foodJson.put("FImage", list.get(i).getfImage());
                foodlist.add(foodJson);
```

```
            }
            response.getWriter().print(new JSONObject().put("FoodList",
            foodlist));
        } catch (JSONException e) {
            //TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

对应的数据库连接类 food_Dao 的代码如下所示。

```
/**
 * 取得全部菜品的信息
 * /
public  ArrayList<foodBean>FoodList(){
    ArrayList<foodBean>list=new ArrayList<foodBean>();
    String sql="SELECT * FROM 'sellingwebsite'.'food';";
    ResultSet rs=db.executeQuery(sql);
    try {
        while (rs.next()) {
            foodBean foodbean=new foodBean();

            foodbean.setfID(rs.getString("fID"));
            foodbean.setfName(rs.getString("fName"));
            foodbean.setfDescription(rs.getString("fDescription"));
            foodbean.setfType(rs.getString("fType"));
            foodbean.setfHobbies(rs.getString("fHobbies"));
            foodbean.setfTechnology(rs.getString("fTechnology"));
            foodbean.setfSeasonal(rs.getString("fSeasonal"));
            foodbean.setfPrice(Double.parseDouble(rs.getString("fPrice")));
            foodbean.setfImage(rs.getString("fImage"));
            foodbean.setfSalesVolume(Integer.parseInt(rs.getString
            ("fSalesVolume")));
            list.add(foodbean);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }finally{
        db.closeDB();
    }
    return list;
}
```

类似于 foodBean 这样的和数据库映射的 Entity 对象的代码在此处省略，具体请参考本章附带的实例代码。

Web 服务器端的 C# 代码如下所示。

```
namespace DAL.Factory
{
    //加载所有的菜品
    public class FoodList:IDAL
    {
        public IJson Execute(FormCollection form)
        {
            IJson json=null;
            string fid=form["Fid"];
            string sqlString=@"Select * from food ";
            if (!string.IsNullOrEmpty(fid))
            {
                sqlString+="where fid="+fid;
            }
            MySqlDataReader reader=SqlHelper.GetExecuteReader(sqlString);
            List<Food>list=new List<Food>();
            while (reader.Read())
            {
                Food temp=new Food();
                temp.FId=reader["fID"].ToString();
                temp.FName=reader["fName"].ToString();
                temp.FDescription=reader["fDescription"].ToString();
                temp.FType=reader["fType"].ToString();
                temp.FHobbies=reader["fHobbies"].ToString();
                temp.FTechnology=reader["fTechnology"].ToString();
                temp.FSeasonal=reader["fSeasonal"].ToString();
                temp.FPrice=double.Parse(reader["fPrice"].ToString());
                temp.FImage=reader["fImage"].ToString();
                temp.FSaleVolume=int.Parse(reader["fSalesVolume"].ToString());

                list.Add(temp);
            }
            reader.Close();
            json=new IJson();
            json.FoodList=list;
            return json;
        }
    }
}
```

同样,关于菜品对象的数据库映射的 Entity 对象的代码在此也省略,具体请参考本章附带的实例代码。

(3) 菜品详情界面的实现。该页面包含两部分的内容,一是菜品的具体信息;二是关于这个菜品的评论。菜品具体信息的 HTML 代码如下所示。

```
<div class="FoodInfo">
    <div class="infoLeft">
        <div class="LikeIt">
```

```
            <a href="#Comment"><span>0</span>条点评</a>
        </div>
        <table class="DetailInfo" border="0" cellspacing="0" cellpadding="0">
            <caption id="FName">

            </caption>
            <tbody>
            <tr>
                <td>菜品类型:</td>
                <td id="FType"></td>
                <td>口味:</td>
                <td id="FHobbies"></td>
            </tr>
            <tr>
                <td>推荐季节:</td>
                <td id="FSeasonal"></td>
                <td>工艺:</td>
                <td id="FTechnology"></td>
            </tr>
            </tbody>
        </table>
        <p id="FDescription"></p>
    </div>
    <div class="infoRight">
        <img id="FImage" src="img/food2.jpg" width="320" style="margin-top:
            10px;" />
        <div style="margin-top: 10px">
            <form id="AddInCart" method="post" data-form-ajax="true">
                <input type="hidden" id="FId" name="fId">
                <input type="submit" style="float: left;
                    margin-left: 20px;background-color: #428bca" value="加入购物
                    车" />
            </form>
            <form method="post">
                <input type="submit" style="float: right;
                    margin-right: 20px;background-color: #f65a5b" value="立即下
                    单" />
            </form>
        </div>
    </div>
</div>
```

加载的方式和菜品预览的方式一样，也是通过 jQuery 的 ajax 发送请求，取得菜品信息之后在前台显示，对应的 JavaScript 代码如下所示。

```
//加载菜品详情
var LoadDetail=function (fid) {
    var options={
        url: "./ajaxQuestServletFactory",
```

```
        type:"GET",
        data: "Fid="+fid+"&ajax="+"FoodDetail",
        dataType: "Json",
        async: false
    };
    $.ajax(options).done(function (obj) {
        var Food=obj.FoodList[0];
        var FName=Food.FName+"<a><span>"+Food.FPrice+"</span></a>";
        $("#FName").html(FName);
        $("#FType").html(Food.FType);
        $("#FHobbies").html(Food.FHobbies);
        $("#FSeasonal").html(Food.FSeasonal);
        $("#FTechnology").html(Food.FTechnology);
        $("#FImage").attr("src", Food.FImage);
        $("#FDescription").html(Food.FDescription);
    })
    return false;
}
```

Web 服务器端的 Java 代码如下所示。

```
/**
 * 加载菜品详情
 * /
public class FoodDetail implements Action {
      public void handle (HttpServletRequest request, HttpServletResponse
      response) throws IOException {
        String Fid=request.getParameter("Fid");
        foodBean food=new food_Dao().FoodList(Fid);
        JSONObject foodJson=new JSONObject();
        ArrayList<JSONObject>FoodList=new ArrayList<JSONObject>();
        try {
            foodJson.put("FId", food.getfID());
            foodJson.put("FName", food.getfName());
            foodJson.put("FDescription",food.getfDescription());
            foodJson.put("FType", food.getfType());
            foodJson.put("FHobbies", food.getfHobbies());
            foodJson.put("FSeasonal", food.getfSeasonal());
            foodJson.put("FTechnology", food.getfTechnology());
            foodJson.put("FPrice", food.getfPrice());
            foodJson.put("FImage", food.getfImage());
            FoodList.add(foodJson);
            response.getWriter().print(new JSONObject().put("FoodList",
            FoodList));
        } catch (JSONException e) {
            //TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

对应的数据库连接类 food_Dao 的代码如下所示。

```
/**
 * 根据菜品 ID 取得菜品详情
 */
public foodBean FoodList(String Fid){
    foodBean foodbean=new foodBean();
    String sql="SELECT * FROM 'sellingwebsite'.'food'
                WHERE 'sellingwebsite'.'food'.'fID'='"+Fid+"';";
    ResultSet rs=db.executeQuery(sql);
    try {
        if(rs.next()) {
            foodbean.setfID(rs.getString("fID"));
            foodbean.setfName(rs.getString("fName"));
            foodbean.setfDescription(rs.getString("fDescription"));
            foodbean.setfType(rs.getString("fType"));
            foodbean.setfHobbies(rs.getString("fHobbies"));
            foodbean.setfTechnology(rs.getString("fTechnology"));
            foodbean.setfSeasonal(rs.getString("fSeasonal"));
            foodbean.setfPrice(Double.parseDouble(rs.getString("fPrice")));
            foodbean.setfImage(rs.getString("fImage"));
            foodbean.setfSalesVolume(Integer.parseInt(rs.getString
            ("fSalesVolume")));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }finally{
        db.closeDB();
    }
    return foodbean;
}
```

Web 服务器端的 C# 代码由于在 FoodList 的 SQL 语句中已经包含了带参数的查询方式，所以这里的代码比较简单，可以直接调用之前定义的方法，具体的代码如下所示。

```
namespace DAL.Factory
{
    public class FoodDetail : IDAL
    {
        public IJson Execute(FormCollection form)
        {
            IJson json=new FoodList().Execute(form);
            return json;
        }
    }
}
```

下面是菜品评论部分的实现，HTML 代码如下所示。

```
<div class="Comment" id="CommentPanel">
    <div class="CommentContainer">
        <hr style="margin-bottom: 20px;" />
        <form method="post" id="AddComment" data-form-ajax="true">
            <div class="CommentDes">
                <span style="display: inline-block;float: left;margin-right:
                20px;">
                    我的点评：</span>
                <input type="hidden" name="account" id="Account"/>
                <input type="hidden" name="fid" id="HiddenFid" />
                <input type="hidden" name="time" id="Time" />
                <textarea name="comInfo" id="ComInfo" rows="5" cols="20"
                      style="width:998px;outline: none;overflow-y: hidden">
                </textarea>
            </div>
            <div style="height: 60px;">
                <input type="submit" id="CommentSubmit" class="submitbtn"
                value="评价"
                    style="width: 250px;height: 40px;margin: 10px 0 10px 100px;
                    float: right;border-radius: 5px; color: #FFF;
                    background-color: #5A9522; border: 1px solid #EEE;
                    outline: none;font-weight: bold;" />
            </div>
        </form>
        <div class="CommentPanel">
        </div>
    </div>
</div>
```

JavaScript 代码如下所示。

```
//验证是否登录
var LoadCookie=function () {
    if($.cookie("User")=="null") {
    $("#ComInfo").html("请登录后再尝试评价")
        $("#ComInfo").attr("readonly", "readonly");
        $("#CommentSubmit").css("background", "#DBDBDB").attr("disabled",
        "disabled")
    }else {
        $("#Account").val($.cookie("User"));
    }
}
    //加载评价
    var LoadComment=function (fid) {
        var options={
            url: "./ajaxQuestServletFactory",
            type: "POST",
            data: "Fid="+fid+"&ajax="+"Comment",
            dataType: "Json",
```

```
            async: false,
            timeout:5000
        };
        $.ajax(options).done(function (obj) {
            var Comments=obj.CommentList;
            //评论条数
            $(".LikeIt>a>span").html(Comments.length);
            $.each(Comments, function (index, values) {
               var ElementImg="<img class='img-circle' src='img/avatar.jpg'
                   width='48'/>";
              var ElementImgA="<a class='pull-left'>"+ElementImg+"</a>";
                var ElementUserName="<h3 class='media-heading'>"+values.
                    BuyerName+"</h3>";
              var ElementDate="<p class='comment-date'>"+values.Time+"</p>";
              var ElementComment="<p class='comment-content'>"+values.Comment+
                  "</p>";
              var ElementBody="<div class='media-body'>"+ElementUserName+
                  ElementDate+ElementComment+"</div>";
              var ElementMedia="<div class='media'>"+ElementImgA+ElementBody+
                  "</div>";
              $(".CommentPanel").append(ElementMedia).append("<hr class='sm'>");
            })
        })
        return false;
    }
```

“评论”一览功能的 Java 代码如下所示。

```
/**
 * 加载菜品评论一览
 * /
public class Comment implements Action {
      public void handle (HttpServletRequest request, HttpServletResponse
      response)
      throws IOException {
        String Fid=request.getParameter("Fid");
        ArrayList<commentBean>list=new comment_Dao().CommentList(Fid);
        ArrayList<JSONObject>CommentList=new ArrayList<JSONObject>();
        try {
            for(int i=0; i<list.size();i++){
                JSONObject comment=new JSONObject();
                comment.put("BuyerName", list.get(i).getBuyerName());
                comment.put("Time", list.get(i).getTime());
                comment.put("Comment", list.get(i).getComment());
                CommentList.add(comment);
            }

        response.getWriter().print(newJSONObject().put("CommentList",
        CommentList));
```

```
        } catch (JSONException e) {
            //TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

对应的 Dao 部分的代码如下所示。

```
/**
 * 根据商品 ID 加载商品评论
 */
public ArrayList<commentBean>CommentList(String Fid){
    ArrayList<commentBean>list=new ArrayList<>();
    String sql="SELECT * FROM 'sellingwebsite'.'comment'
                WHERE 'sellingwebsite'.'comment'.'fID'='"+Fid+"';";
    ResultSet rs=db.executeQuery(sql);
    try {
        while (rs.next()) {
            commentBean commentbean=new commentBean();
            commentbean.setcID(Integer.parseInt(rs.getString("cID")));
            commentbean.setfID(rs.getString("fID"));
            commentbean.setBuyerName(rs.getString("buyerName"));
            commentbean.setComment(rs.getString("comment"));
            commentbean.setTime(rs.getString("time"));
            list.add(commentbean);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }finally{
        db.closeDB();
    }
    return list;
}
```

添加评论的 Java 代码如下所示。

```
/**
 * 添加菜品评论
 */
public class AddComment implements Action {
    public void handle(HttpServletRequest request, HttpServletResponse
    response)
    throws IOException {
        //取得待评论菜品的 ID 和评论人员的身份信息
        String account=request.getParameter("account");
        String fid=request.getParameter("fid");
        String time=new comment_Dao().int_data_time();
        String comInfo=request.getParameter("ComInfo");
```

```
        urlBean urlbean=new urlBean();
        JSONObject obj=new JSONObject();
        commentBean commentbean=new commentBean();
        commentbean.setBuyerName(account);
        commentbean.setfID(fid);
        commentbean.setTime(time);
        commentbean.setComment(comInfo);
        int i=new comment_Dao().addComment(commentbean);
        //处理结果
        if(i>=1) {
            urlbean.setJsonType("AddComment");
            urlbean.setIsSuccess(true);
        } else {
            urlbean.setJsonType("AddComment");
            urlbean.setIsSuccess(false);
        }

        //向前端返回 json 数据
        try {
            obj.put("JsonType", urlbean.getJsonType());
            obj.put("IsSuccess", urlbean.getIsSuccess());
        } catch (JSONException e) {
            e.printStackTrace();
        }
        response.getWriter().print(obj);
    }
}
```

对应的 comment_Dao 代码如下所示。

```
/**
  * 添加评论
  * /
public int addComment(commentBean commentbean){
    String sql="INSERT INTO 'sellingwebsite'.'comment' ('fID', 'buyerName',
        'comment', 'time')"+" VALUES ('"+commentbean.getfID()+"',
        '"+commentbean.getBuyerName()+"',
        '"+commentbean.getComment()+"', '"+commentbean.getTime()+"');";
    int r=db.executeUpdate(sql);
    db.closeDB();
    return r ;
}
```

评论预览功能的 C#代码如下所示。

```
namespace DAL.Factory
{
    public class Comment : IDAL
    {
        public IJson Execute(FormCollection form)
```

```
        {
            IJson json=null;
            string fid=form["Fid"];

            string sqlString=@"Select * from Comment where fid="+fid;

            MySqlDataReader reader=SqlHelper.GetExecuteReader(sqlString);
            List<Comt>list=new List<Comt>();
            while (reader.Read())
            {
                Comt temp=new Comt();
                temp.CId=reader["cID"].ToString();
                temp.FID=reader["fID"].ToString();
                temp.BuyerName=reader["buyerName"].ToString();
                temp.Comment=reader["comment"].ToString();
                temp.Time=reader["time"].ToString();
                list.Add(temp);
            }
            reader.Close();
            json=new IJson();
            json.CommentList=list;

            return json;
        }
    }
}
```

添加评论的 C# 代码如下所示。

```
namespace DAL.Factory
{
    public class AddComment : IDAL
    {
        public IJson Execute(FormCollection form)
        {
            IJson json=null;

            string fid=form["fid"];
            string buyerName=form["account"];
            string comment=form["comInfo"];
            string time=form["time"];

            string sqlString=@"insert into comment(fid,buyerName,comment,time)
                             values(@fID,@buyerName,@comment,@time)";
            MySqlParameter[] para=new MySqlParameter[]
            {
                new MySqlParameter("@fID",fid),
                new MySqlParameter("@buyerName",buyerName),
                new MySqlParameter("@comment",comment),
```

```
                new MySqlParameter("@time",DateTime.Parse(time))
            };
            int result=SqlHelper.GetExecuteNonQuery(sqlString, para);
            if (result !=-1)
            {
                json=new IJson("AddComment", true, null);
            }
            return json;
        }
    }
}
```

8.3.3　注册和登录

顾客在首页和菜品详情页面了解到菜品的具体信息且有购买意向的时候，需要用户登录后才能锁定到具体的送餐地址等信息。

(1) 网页设计。界面的风格上保持了和首页一样的风格，具体的界面设计如图 8-7 和图 8-8 所示。



图 8-7　用户注册



图 8-8　用户登录

（2）注册和登录功能的实现。注册和登录功能是每个系统中必备的功能，在这里只展示 HTML 代码部分，其他后台代码部分各位读者可以自行完成。注册页面的 HTML 代码如下所示。

```
<div class="Loginbanner animated fadeInUp">
    <div class="LoginContainer animated InRightBig">
        <div class="LoginPanel animated">
            <h2 style="text-align: center;margin-bottom: 30px;">用户登录
            </h2>
            <form method="post" id="Login" data-form-ajax="true">
                <div class="form-input">
                    <input name="account" id="LoginAccount" required
                    autocomplete="off"
                      class="form-control" />
                      <label class="input-lable">用户名</label>
                </div>
                <div class="form-input">
                    <input name="password" required type="password" class=
                    "form-control" />
                    <label class="input-lable">密  码</label>
                </div>
                <div class="floating-btn">
                    <i class="icon-arrow"></i>
                    <input type="submit" value="" style="opacity: 0" />
                </div>
                <div class="validate-Login animated">
                    <!--用户验证提示-->
                    <span class="validate-Auth-Login"></span>
                </div>
                <div style="margin-top: 10px">
                    <a class="RegBtn">免费注册？</a>
                    <label style="float: right">
                      <input type="checkbox" style="margin-right: 5px;">记住登录
                    </label>
                </div>
            </form>
        </div>
        <div class="RegPanel animated" style="display: none">
            <h2 style="text-align: center;margin-bottom: 30px;">账户注册
            </h2>
            <form method="post" id="Register" data-form-ajax="true">
                <div class="form-input">
                    <input required="required" name="account" id="RegAccount"
                        autocomplete="off" type="text" class="form-control" />
                    <label class="input-lable">用户名</label>
                </div>
                <div class="form-input">
                    <input required="required" name="password" type="password"
                        class="form-control" />
```

```
                <label class="input-lable">密　码</label>
            </div>
            <div class="form-input">
                <input required="required" name="confirmPass" type=
                "password"
                    class="form-control" />
                <label class="input-lable">确认密码</label>
            </div>
            <div class="form-input">
                <input required="required" name="address" autocomplete="off"
                    type="text" class="form-control" />
                <label class="input-lable">地址</label>
            </div>
            <div class="form-input">
                <input required="required" name="tel" autocomplete="off"
                    maxlength="11" type="tel" class="form-control" />
                <label class="input-lable">联系电话</label>
            </div>
            <div class="floating-btn">
                <i class="icon-arrow"></i>
                <input type="submit" value="" style="opacity: 0" />
            </div>
        </form>
        <div class="validate-Reg animated">
            <span class="validate-Auth-Reg"></span>
        </div>
        <div style="margin-top: 10px;">
            <a class="LoginBtn">已有账号,登录</a>
        </div>
    </div>
  </div>
</div>

<script>
    $(function () {
        var thisURL=document.URL;
        var para=thisURL.split('?')[1];
        var key=para.split("=")[1];
        if (key=='false') {
            $(".LoginPanel").fadeOut(1)
            $(".RegPanel").fadeIn(1).addClass('flipInX');
        }
    })
</script>
```

主要的 CSS 代码如下所示。

```
.Loginbanner {
    width: 100%;
    z-index: 1;
}

.LoginContainer {
    margin: 0 auto;
    width: 1200px;
    padding-top: 100px;
    right: 20px;
    position: relative;
}

.LoginPanel {
    /* background-color: #FF4081; */
    background-color: #FFF;
    width: 340px;
    height: 230px;
    position: absolute;
    right: 30px;
    border-radius: 10px;
    padding: 45px 20px;
}

.RegBtn:hover,
.LoginBtn:hover {
    cursor: pointer;
    text-decoration: underline;
    color: #FF4081;
}

.RegPanel {
    background-color: #FFF;
    width: 340px;
    height: 440px;
    z-index: 99;
    position: absolute;
    right: 30px;
    border-radius: 10px;
    padding: 45px 20px;
}
```

后台的 Java 和 C# 代码见本章附带的实例代码。

8.3.4 购物车

在菜品一览界面和菜品详情界面，系统可以提供添加到购物车的功能。购物车功能作为购物网站的必备功能，具有将用户感兴趣的商品收纳其中的作用。

（1）网页设计。本系统中，为了使购物车的页面和订单页面有所区别，只提供了一个简

易的购物车。该功能在鼠标悬停于导航栏的购物车菜单时直接下拉弹出，还提供了菜品删除和立即支付的功能，设计图如图 8-9 所示。



图 8-9　购物车

（2）购物车界面的实现。购物车的 HTML 代码在导航栏部分的实现中已经介绍过。取得个人购物车信息的 JavaScript 代码如下所示。

```
//加载购物车预览功能
var LoadShopCar=function () {
    $("#CarProcess").show("normal");
    var options={
        url: "./ajaxQuestServletFactory",
        type: "POST",
        data: "account="+$.cookie("User")+"&ajax="+"ShopCart",
        dataType: "Json"
    };
    $.ajax(options).done(function (obj) {
        var Foods=obj.FoodList;
        $.each(Foods, function (index, values) {
            var ElementHidden="<input type='hidden' name='cartId' value='"+
                values.FId+"'/>";
            var ElementSubmit="<span class='dBtn'></span>"
            var ElementForm="<div>"+ElementHidden+ElementSubmit+"</div>";

            var ElementSpan="<span>"+values.FName+"</span>";
            var ElementImg="<img src='"+values.FImage+"'/>";
            var ElementA="<a href='detail.html?id="+values.FId+"'>"+
                ElementImg+ElementSpan+"</a>";
            var ElementAdiv="<div>"+ElementA+"</div>";
            var ElementLi="<li>"+ElementAdiv+ElementForm+"</li>";
            $("#CarList").append(ElementLi);
        })
        $("#CarProcess").hide("normal");
    })
    return false;
}
```

```
    //加载购物车事件
    $(".carBtn").click(function () {
        LoadShopCar();
    })
    //删除购物车内的菜品
    $("#CarList").delegate(".dBtn", "click", function () {
        var i=$(this);
        var href=i.prev('input').val()
        var options={
            url: "./ajaxQuestServletFactory",
            type: "POST",
            data: "fid="+href+"&account="+$.cookie("User")+"&ajax="+
            "DeleteInCart", dataType: "Json"
        };
        $.ajax(options).done(function (obj) {
            if (Boolean(obj.IsSuccess)) {
                i.parent().parent().remove();
            }
        })
    })
```

对应购物车的 Java 代码如下所示。

```
/**
 * 加载购物车
 */
public class ShopCart implements Action {
    public void handle(HttpServletRequest request, HttpServletResponse
    response)
    throws IOException {
        String account=request.getParameter("account");
        ArrayList<cartBean>list=new Cart_Dao().CartList(account);
        ArrayList<JSONObject>cartlist=new ArrayList<JSONObject>();
        try {
            for(int i=0; i<list.size();i++){
                JSONObject cartJson=new JSONObject();

                cartJson.put("FId", list.get(i).getfID());
                cartJson.put("FName", list.get(i).getfName());
                cartJson.put("FImage", list.get(i).getfImage());
                cartlist.add(cartJson);
            }
            response.getWriter().print(new JSONObject().put("FoodList",
            cartlist));
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

```
/**
 * 删除购物车控制器
 * /
public class DeleteInCart implements Action {
    public void handle(HttpServletRequest request, HttpServletResponse
    response)
    throws IOException {
        String account=request.getParameter("account");
        String fid=request.getParameter("fid");
        int i=new Cart_Dao().deleteCart(fid, account);
        try {
            if(i>=1){
                response.getWriter().print(new JSONObject().put("IsSuccess",
                true));
            }else{
                response.getWriter().print(new JSONObject().put("IsSuccess",
                false));
            }
        } catch(JSONException e) {
            e.printStackTrace();
        }
    }
}
```

对应的数据库操作部分的 Dao 对应的代码如下所示。

```
/**
 * 加载商品预览功能
 * /
public ArrayList<cartBean>CartList(String account){
    ArrayList<cartBean>list=new ArrayList<cartBean>();
    String sql="SELECT * FROM 'sellingwebsite'.'shop_cart'
                WHERE 'buyerName'='"+account+"';";
    ResultSet rs=db.executeQuery(sql);
    try {
        while (rs.next()) {
            cartBean cartbean=new cartBean();
            cartbean.setfID(rs.getString("fID"));
            cartbean.setBuyerName(rs.getString("buyerName"));
            cartbean.setfName(rs.getString("fName"));
            cartbean.setfImage(rs.getString("fImage"));
            list.add(cartbean);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }finally{
        db.closeDB();
```

```
    }
    return list;
}

/**
 * 删除购物车内的菜品
 * /
public int deleteCart(String fid,String account){
    String sql ="DELETE FROM 'sellingwebsite'.'shop_cart' WHERE "
            +sellingwebsite'.'shop_cart'.'buyerName'='"+account+"' AND
            'sellingwebsite'.'shop_cart'.'fID'='"+fid+"';";
    int i=db.executeUpdate(sql);
    db.closeDB();
    return i;
}
```

购物车 C# 的代码如下所示。

```
namespace DAL.Factory
{
    //取得购物车预览功能
    public class ShopCart : IDAL
    {
        public IJson Execute(FormCollection form)
        {
            IJson json=null;
            string account=form["account"];
            string sqlString=@"select * from shop_cart where buyerName=
            @account";
            MySqlParameter para=new MySqlParameter("@account", account);

            MySqlDataReader reader=SqlHelper.GetExecuteReader(sqlString,
            para);
            List<Food>list=new List<Food>();
            while (reader.Read())
            {
                Food temp=new Food();
                temp.FId=reader["fID"].ToString();
                temp.FName=reader["fName"].ToString();
                temp.FImage=reader["FImage"].ToString();
                list.Add(temp);
            }
            reader.Close();
            json=new IJson();
            json.FoodList=list;
            return json;
        }
    }
}
```

```
namespace DAL.Factory
{
    public class DeleteInCart : IDAL
    {
        ///删除购物车内的菜品
        public IJson Execute(FormCollection form)
        {
            IJson json=new IJson();
            string account=form["account"];
            string fid=form["fid"];
            string sqlString=@"delete from shop_cart where fId=@fid
                                and buyerName=@buyerName ";

            MySqlParameter[] para=new MySqlParameter[]
            {
                new MySqlParameter("@fid",fid),
                new MySqlParameter("@buyerName",account)
            };
            int result=SqlHelper.GetExecuteNonQuery(sqlString, para);
            if (result !=-1)
            {
                json.IsSuccess=true;
            }
            else
            {
                json.IsSuccess=false;
            }
            return json;
        }
    }
}
```

8.3.5　订单管理

用户下单之后的菜品会进入订单页面，订单功能也是商城类网站中必不可少的功能，记录了顾客的购买数据，是顾客购买的凭证，也是商家分析菜品销量的重要数据。

(1) 网页设计。用户端的订单页面只提供了简单的展示功能，具体的界面如图 8-10 所示。



图 8-10　用户订单界面

(2) 订单预览界面的实现。该页面的加载方式和菜品预览界面的一样，也是采用在页面加载完成的时候向服务器发送 ajax 请求来实现的，具体的 HTML 代码如下所示。

```
<div class="NoOrder"
    style="margin: 25px auto;width: 1200px;line-height: 30px;
        text-align: center;display: none;color:#777;display:none">
    <p>暂无订单,快去下单吧!
    <a href="menu.html"  style="background-color: #5A9522;color: #FFF;
            line-height: 30px;border-radius: 5px;
            display: inline-block;padding: 0 10px;float: right">下单去</a>
    </p>
</div>

<div class="Order" style="display:none;">
    <div class="OrderContainer">
        <div class="caption animated InDownBig" style="animation-duration:
        .5s;">
            <ul>
                <li>图片</li>
                <li>名称</li>
                <li>单价</li>
                <li>数量</li>
                <li>实际付款</li>
                <li>交易状态</li>
            </ul>
        </div>
    </div>
</div>
```

对应的 JavaScript 代码如下所示。

```
//加载详情订单
var LoadOrder=function () {
    var options={
        url: "./ajaxQuestServletFactory",
        type: "Post",
        data: "account="+$.cookie("User")+"&ajax="+"Order",
        dataType: "Json",
        async: false
    };
    $.ajax(options).done(function (obj) {
        console.log(obj.OrderList)
            var Orders=obj.OrderList;
            if (Orders !="") {
                $(".NoOrder").css("display", "none");
                $.each(Orders, function (index, values) {
                    var img="<img src='"+values.FImage+"'>";
                var toDetaila="<a class='pull-left animated fadeInLeft'
                                href='detail.html?fid="+values.FId+"'>"+
                                img+"</a>";
```

```
                var li1="<li class='orderDes'><a href='detail.html?fid=
                        "+values.FId+"'>"+values.FName+"<p>"+values.
                        FDescription+"</p></a>
                        <div style='margin-top: 5px'>
                        <a class='oderDetail'>订单详情</a>
                        </div></li>"
                var li2="<li class='priceSpan'><span class='FPrice'>"+
                        values.FPrice+"</span></li>";
                var li3="<li class='orderCount'><span>1</span></li>"
                var li4="<li class='orderSum'><span class='PayMoney'>"+
                        values.PayMoney+"</span></li>";
                var TradingStatus=(values.TradingStatus==1) ? "交易成功" : "交易
                    失败";
                var li5="<li class='liPay'><span class='payState'>"+
                        TradingStatus+"</span></li>"
                var DeliveryStatus=(values.DeliveryStatus==1) ? '已发货' : "备货
                    中";
                var p4="<p><span class='SendState'>"+DeliveryStatus+"</span></p
                       ><p><span class='ReceviceAdd'>"+values.Address+
                       "</span></p><p><span class='Recevicer'>"+values.
                       UserName+"</span></p><p><span class='ReceviceTel'>"+
                       values.Tel+"</span></p>"
                var pdiv="<div class='btnToOrder animated'>"+p4+"</div>";
                var ul="<ul class='fadeInDown animated'>"+
                       li1+li2+li3+li4+li5+pdiv+"</ul>"
                 var funComment="<div><input type='hidden' value='"+values.
                     OrderId+"'>
                     <input type = ' submit ' class = ' deleteOrderBtn animated
                     fadeInRight' value='删除'>
                     <a class='fadeIn animated' style='animation-duration:2s'
                     href='detail.html?fid="+values.FId+"'>评价</a></div>"
                var OrderList="<div class='OrderList'>"+toDetaila+ul+
                              funComment+"</div>";
                var hr="<hr class='sm' />"
                $(".OrderContainer").append(OrderList).append(hr);
            })
            $(".Order").show("normal")
        }else {
            $(".NoOrder").show("normal")
            $(".Order").css("display", "none");
        }
    })
    return false;
}
```

加载订单预览功能的 Java 代码如下所示。

```
/**
 * 加载订单预览功能
 * /
public class Order implements Action {
    public void handle(HttpServletRequest request, HttpServletResponse
    response)
        throws IOException {
            String account=request.getParameter("account");
            ArrayList<orderBean>list=new order_Dao().OrderList(account);
            ArrayList<JSONObject>OrderList=new ArrayList<JSONObject>();
            try {
                for(int i=0; i<list.size();i++){
                    JSONObject order=new JSONObject();
                    order.put("FImage", list.get(i).getfImage());
                    order.put("FId", list.get(i).getfID());
                    order.put("FName", list.get(i).getfName());
                    order.put("FDescription", list.get(i).getfDescription());
                    order.put("FPrice", list.get(i).getfPrice());
                    order.put("PayMoney", list.get(i).getPayMoney());
                    order.put("TradingStatus", list.get(i).getTradingStatus());
                    order.put("DeliveryStatus", list.get(i).getDeliveryStatus());
                    order.put("Address", list.get(i).getAddress());
                    order.put("UserName", list.get(i).getUserName());
                    order.put("Tel", list.get(i).getTel());
                    order.put("OrderId", list.get(i).getOrderID());
                    OrderList.add(order);
                }
                 response.getWriter().print(new JSONObject().put("OrderList",
                 OrderList));
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
}
```

对应的 order_Dao 部分的代码如下所示。

```
/**
 * 根据用户账号加载订单
 * /
public ArrayList<orderBean>OrderList(String account){
    ArrayList<orderBean>list=new ArrayList<orderBean>();
    String sql="SELECT * FROM 'sellingwebsite'.'orders'
                WHERE 'sellingwebsite'.'orders'.'buyerName'='"+account+"';";
    ResultSet rs=db.executeQuery(sql);
    try {
        while(rs.next()) {
            orderBean orderbean=new orderBean();
```

```
                orderbean.setOrderID(Integer.parseInt(rs.getString("orderID")));
                orderbean.setfID(rs.getString("fID"));
                orderbean.setBuyerName(account);
                orderbean.setfName(rs.getString("fName"));
                orderbean.setfDescription(rs.getString("fDescription"));
                orderbean.setfPrice(Double.parseDouble(rs.getString("fPrice")));
                orderbean.setfImage(rs.getString("fImage"));
                orderbean.setPayMoney(Double.parseDouble(rs.getString(
                "payMoney")));
                orderbean.setTradingStatus(Integer.parseInt(rs.getString(
                "tradingStatus")));
                orderbean.setDeliveryStatus(Integer.parseInt(rs.getString(
                "deliveryStatus")));
                orderbean.setTime(rs.getString("time"));
                orderbean.setUserName(rs.getString("userName"));
                orderbean.setAddress(rs.getString("address"));
                orderbean.setTel(rs.getString("tel"));
                list.add(orderbean);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }finally{
            db.closeDB();
        }
        return list;
}
```

在 C# 中的后台服务器端的代码如下所示。

```
namespace DAL.Factory
{
    class Order : IDAL
    {
        public IJson Execute(FormCollection form)
        {
            IJson json=null;
            string account=form["account"];
            string sqlString=@"select * from orders where buyerName=
            '"+account+"'";

            MySqlDataReader reader=SqlHelper.GetExecuteReader(sqlString);
            List<Od>list=new List<Od>();
            while (reader.Read())
            {
                Od temp=new Od();
                temp.OrderId=reader["orderID"].ToString();
                temp.FId=reader["fID"].ToString();
                temp.BuyerName=reader["buyerName"].ToString();
                temp.FName=reader["fName"].ToString();
```

```
                temp.FDescription=reader["fDescription"].ToString();
                temp.FPrice=double.Parse(reader["fPrice"].ToString());
                temp.FImage=reader["fImage"].ToString();
                temp.PayMoney=reader["payMoney"].ToString();
                temp.TradingStatus=int.Parse(reader["tradingStatus"].
                ToString());
                temp.DeliveryStatus=int.Parse(reader["deliveryStatus"].
                ToString());
                temp.Time=DateTime.Parse(reader["time"].ToString());
                temp.UserName=reader["userName"].ToString();
                temp.Address=reader["address"].ToString();
                temp.Tel=reader["tel"].ToString();
                list.Add(temp);
            }
            reader.Close();
            json=new IJson();
            json.OrderList=list;
            return json;
        }
    }
}
```

本章小结

本章用之前学到的客户端编程交互式技术完成了一个简易的网上订餐系统。这个系统与用 JSP 或者 ASPX 所开发的项目最大的不同点是采用了异步提交的方式来和服务器进行交互处理。同时，为了满足 Java 和 C# 的服务器端语言，本章对请求采用了集中控制分发的设计，用到了工厂模式。定义了请求和处理类的 XML 文件，能在两种语言中自由地使用，也限定了在不同语言中的类名的命名必须保持一致。关于这个项目的后台管理程序没有在本章中做讲解，希望读者能根据这样的设计，并结合本章的代码实例，自行完成在线订餐系统的后台管理功能模块。

参 考 文 献

[1] 党建. Web 前端开发最佳实践[M]. 北京：机械工业出版社，2015.

[2] Jon Duckett. JavaScript & jQuery 交互式 Web 前端开发[M]. 杜伟，柴晓伟，涂曙光，译. 北京：清华大学出版社，2015.

[3] 刘心美，陈义辉，韩宝玉. Web 前端设计与制作——HTML＋CSS＋jQuery[M]. 北京：清华大学出版社，2016.

[4] 李鸿君，陈品华. Web 前端项目开发实践教程[M]. 武汉：武汉大学出版社，2016.

[5] Patrick McNeil. 网页设计创意书[M]. 石华耀，译. 北京：人民邮电出版社，2014.

[6] Jon Duckett. Web 设计与前端开发秘籍[M]. 杜伟，柴晓伟，涂曙光，译. 北京：清华大学出版社，2015.

[7] Jeremy Keith，Jeffrey Sambells. JavaScript DOM 编程艺术[M]. 2 版. 杨涛，等，译. 北京：人民邮电出版社，2011.

[8] Phil Ballard，Michiel Moncur. JavaScript 入门经典[M]. 5 版. 王军，译. 北京：人民邮电出版社，2013.

[9] Elisabeth Robson，Eric Freeman. Head First HTML 与 CSS＋JavaScript DOM 编程艺术[M]. 2 版. 徐阳，译. 北京：中国电力出版社，2013.

[10] 单东林，张晓菲，魏然. 锋利的 jQuery[M]. 2 版. 北京：人民邮电出版社，2012.

[11] Adam Freeman. 精通 jQuery[M]. 2 版. 魏忠，译. 北京：人民邮电出版社，2014.

[12] 王震江，马宏. XML 基础与 ajax 实践教程[M]. 2 版. 北京：清华大学出版社，2016.

[13] 本社. Web Services 应用开发[M]. 北京：电子工业出版社，2011.

[14] 王石磊. Java Web 开发技术详解[M]. 北京：清华大学出版社，2014.